PW SKILLS

2D Arrays & ArrayLists

Today’s checklist

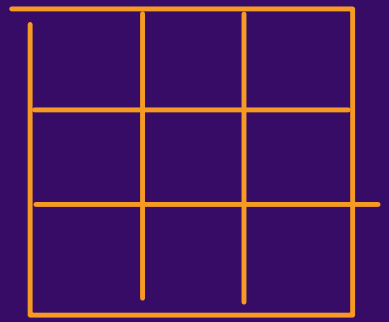
1. Basic of 2D Arrays
2. Problems based on 2D Arrays
3. 2D ArrayList
4. ArrayList vs Arrays
5. Basic STL functions of ArrayList
6. Problem based on 2-D ArrayList

What and Why?

Array → collection of same data type

→ Linear List

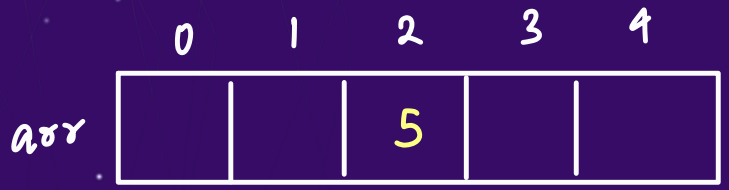
2D Array → is a grid, it is matrix



Representation of 2D array

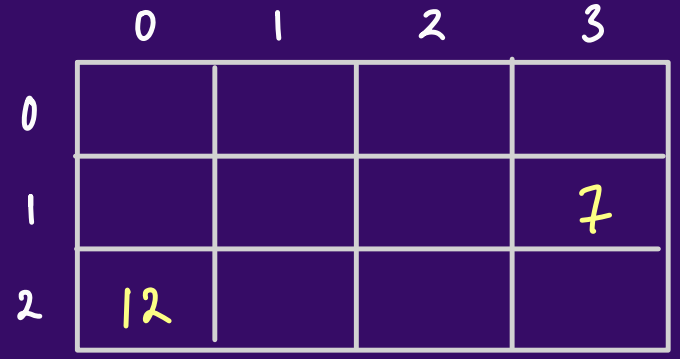
PW SKILLS

```
int[] arr = new int[5];
```




```
arr[2] = 5;
```

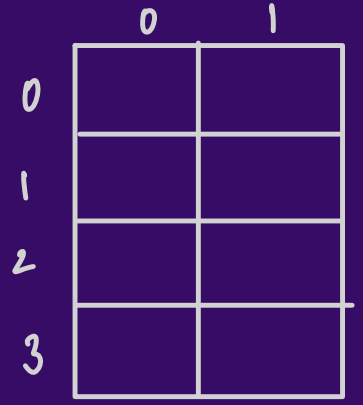
```
int[][] arr = new int[3][4];
```




```
arr[1][3] = 7
arr[2][0] = 12
```

Declaration of a 2-Dimensional Array

```
int[][] arr = new int[4][2];
```

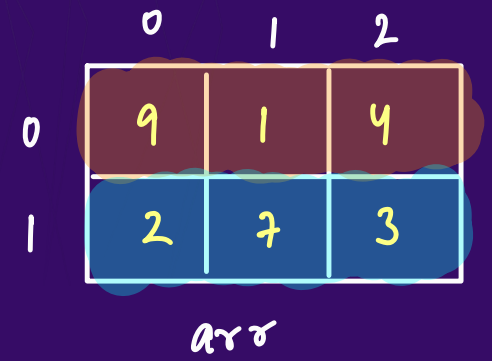



```
int[][] arr = {{1,2},{3,4},{5,6},{7,8}};
```

In Java, in 2D Arrays, if we are directly initializing it, then we do not mention the no. of rows and columns, but if we are only declaring and allocating the memory, then we need to mention both the rows and columns.

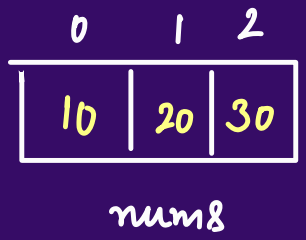
Array of arrays

```
int[][] arr = new int[2][3];
```



arr = { {9, 1, 4}, {2, 7, 3} }

```
int[] nums = new int[3];
```



{10, 20, 30}

Initialisation of a 2-Dimensional Array

```
int[][] arr = { { 1234, 56 }, { 1256, 43 }, { 1434, 32 }, { 1312, 96 } } ;
```

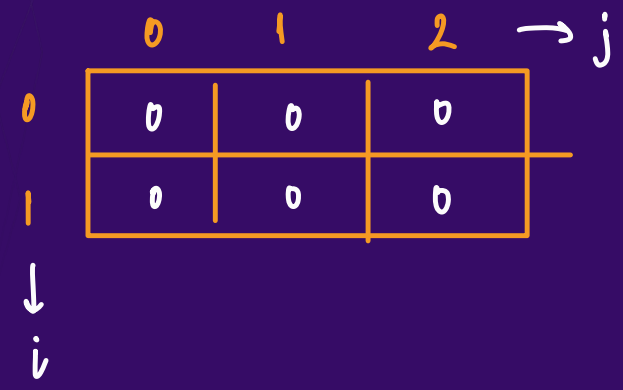
```
int[][] arr = { {12, 34, 56}, { 78, 91, 23} } ;
```

	0	1
0	1234	56
1	1256	43
2	1434	32
3	1312	96

	0	1	2
0	12	34	56
1	78	91	23

Traversal through 2D array

```
int[][] arr = new int[2][3];
```



Taking 2D array as input from the user

PW SKILLS

Ques:

Q1 : Write a program to store roll number and marks obtained by 4 students side by side in a matrix.

	Raghav	Harsh	Sanket	Gagan
Rno	76	88	82	13
Marks	92	87	98	94

	0	1	
0	76	92	Raghav
1	88	87	Harsh
2	82	98	Sanket
3	13	94	Gagan
	Rno	Marks	

Ques:

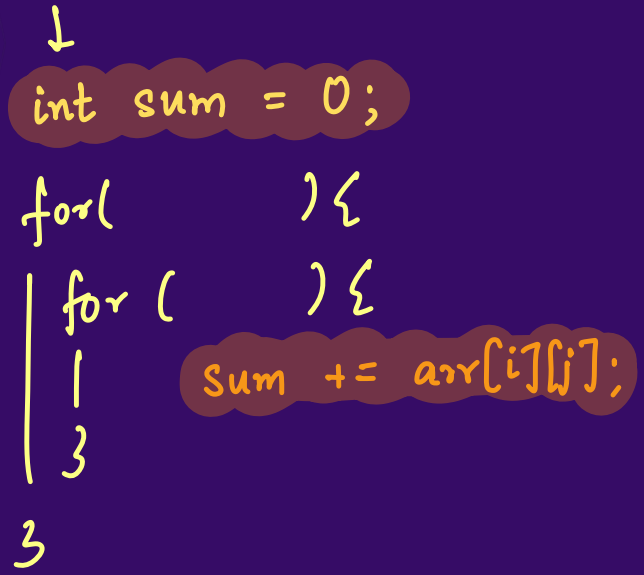
PW SKILLS

Q2 : Write a JAVA program to find the largest element of a given 2D array of integers.

```
int mx = Integer.MIN_VALUE;

for(int i=0 ; i<m; i++){
    for(int j=0 ; j<n; j++){
        mx = Math.max(mx,arr[i][j]);
    }
}
```

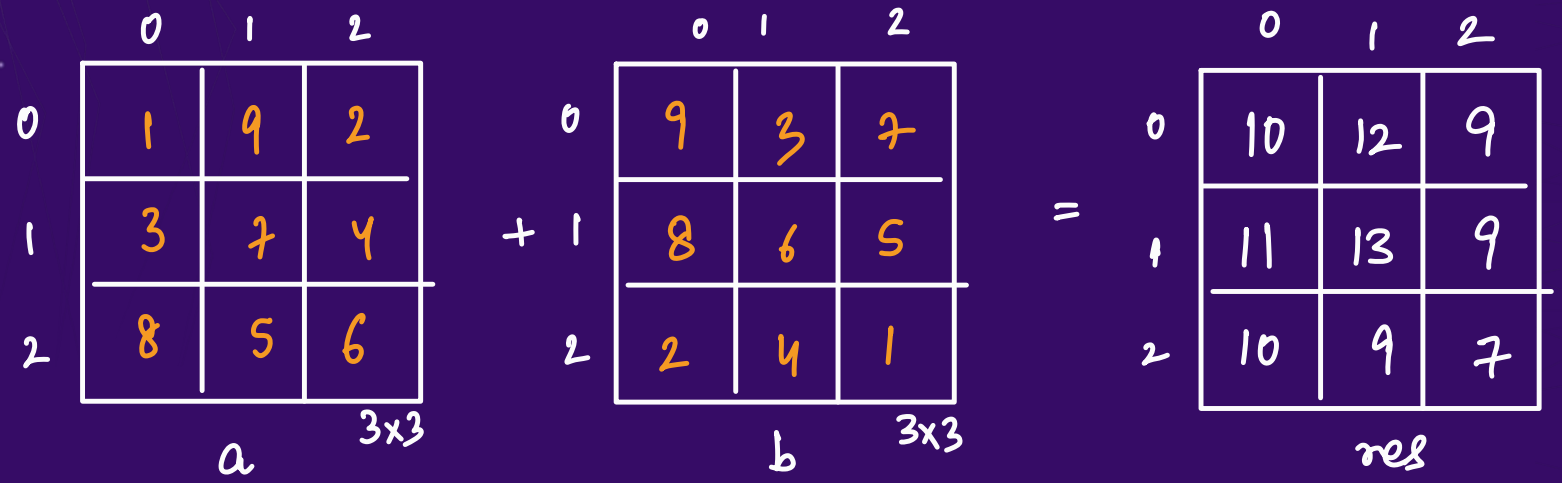
Ques:

Q3 : Write a program to print sum of all the elements of a 2D matrix.



Ques:

Q4 : Write a program to add two matrices.

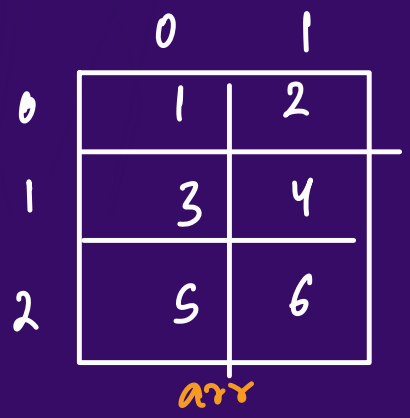



```
res[i][j] = a[i][j] + b[i][j];
```

Ques:

PW SKILLS

Q5 : Write a program to print the transpose of the matrix entered by the user and store it in a new matrix.



Row - Wise Printing

1 2
3 4
5 6

1 2 3 4 5 6

Col wise Printing

1 3 5
2 4 6

→ transpose

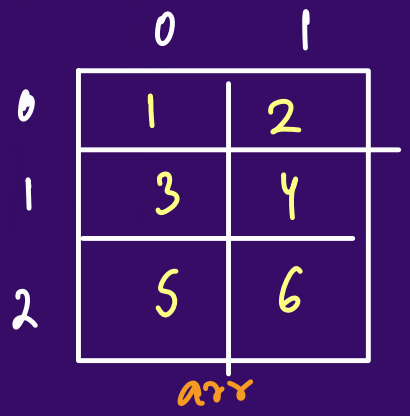
1 3 5 2 4 6

arr = { {1,2}, {3,4}, {5,6} }

Ques:

PW SKILLS

Q5 : Write a program to print the transpose of the matrix entered by the user and store it in a new matrix.

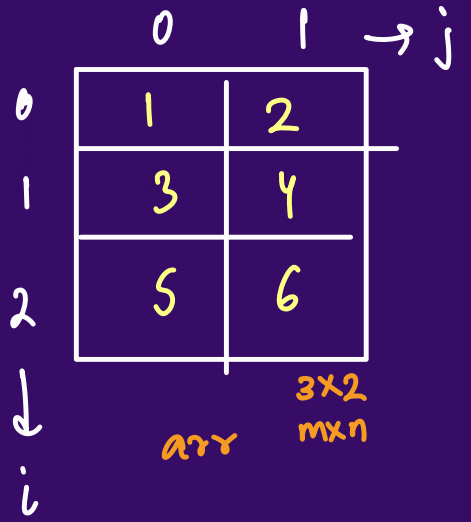


If we have to print for Ex the 0th col,

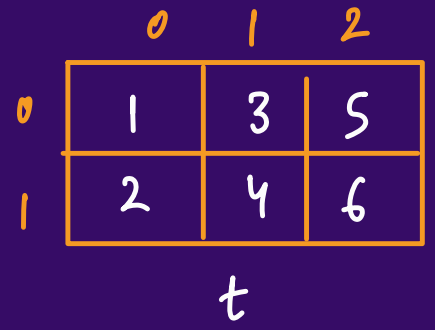
arr[0][0], arr[1][0], arr[2][0]

Ques:

Q5 : Write a program to print the transpose of the matrix entered by the user and store it in a new matrix.



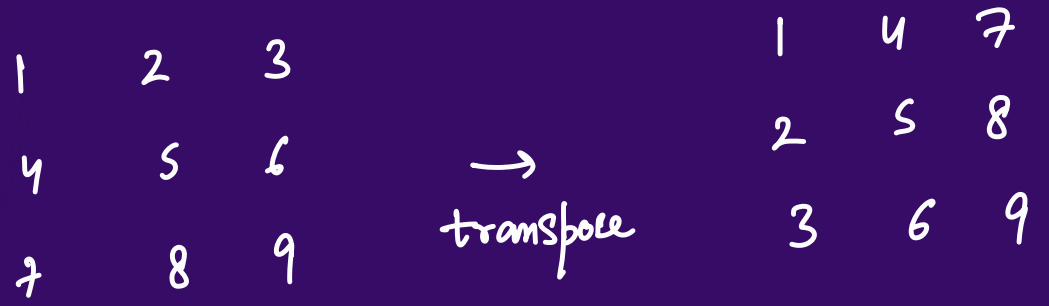

```
int[] transpose = new int[n][m];
```




```
t[i][j] = arr[j][i]
```

Ques:

Q5 : Write a program to print the transpose of the matrix entered by the user and store it in a new matrix.



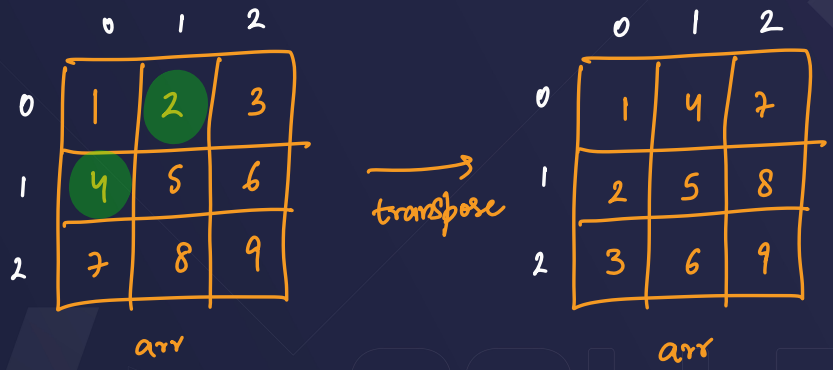
Ques:

PW SKILLS

Q6 : Write a program to change the given matrix with its transpose.

[Leetcode 867]

Square Matrix




```
swap(arr[i][i], arr[i][i]);
```

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12
3	13	14	15	16

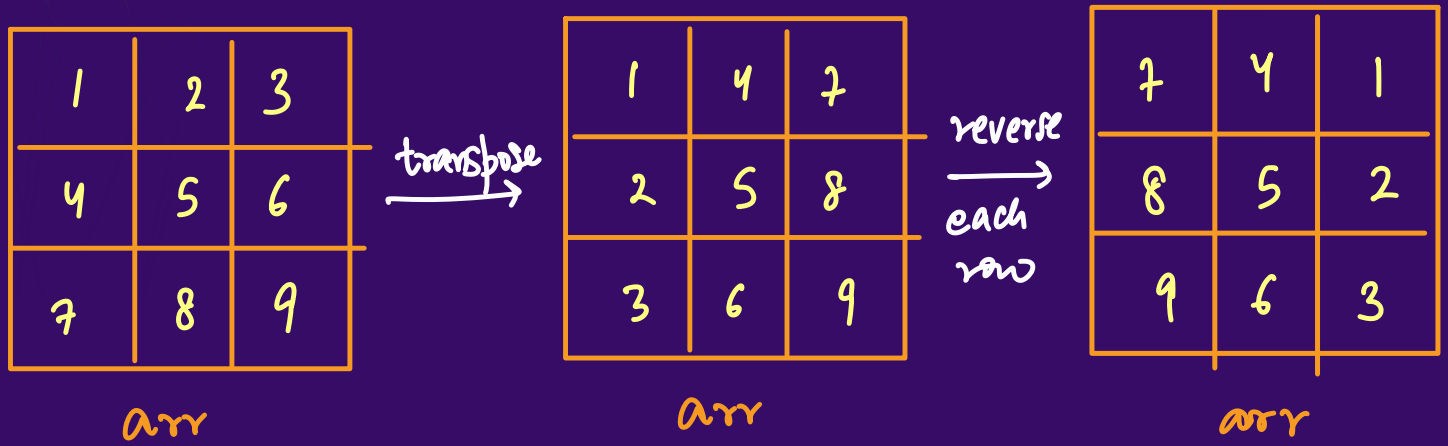
swap (arr[i][j], arr[j][i])

but swap(j, i → i, j

Ques:

PW SKILLS

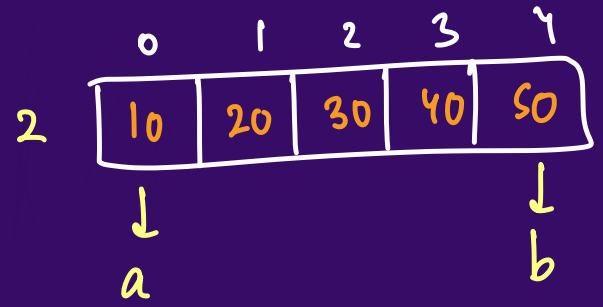
Q7 : Write a program to rotate the matrix by 90 degrees clockwise.



[Leetcode 48]

Ques:

Q7 : Write a program to rotate the matrix by 90 degrees clockwise.



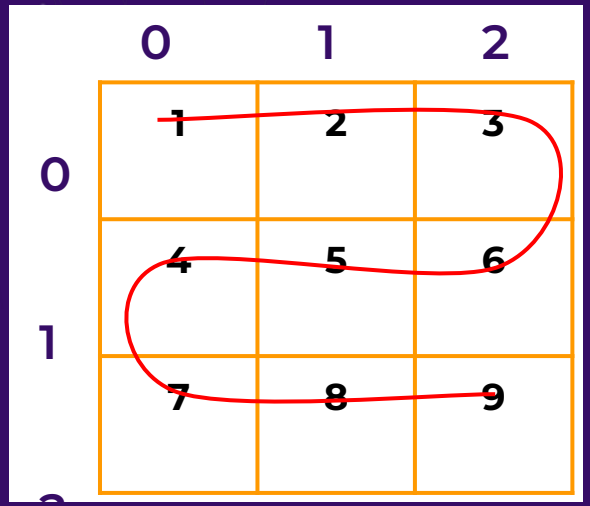

```
swap (arr[i][a], arr[i][b])
    a++; b--;
```

[Leetcode 48]

Ques:

PW SKILLS

Q8 : Write a program to print the matrix in wave form.



Row wise

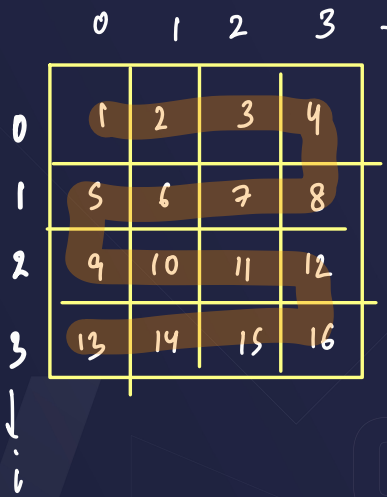
1 2 3 4 5 6 7 8 9

Col wise

1 4 7 2 5 8 3 6 9

wave form (Row wise alternate)

1 2 3 6 5 4 7 8 9

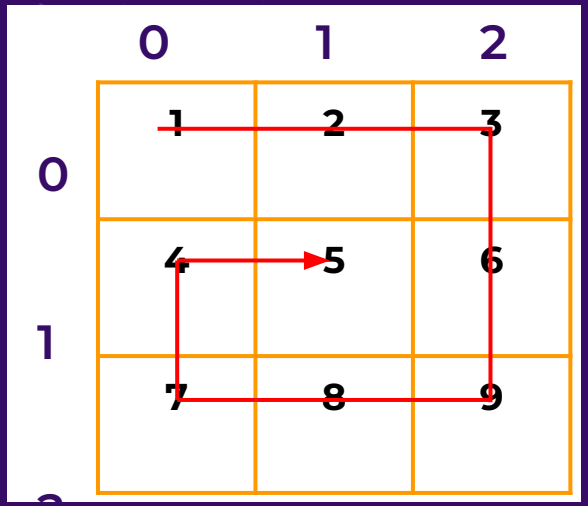


1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

Ques:

PW SKILLS

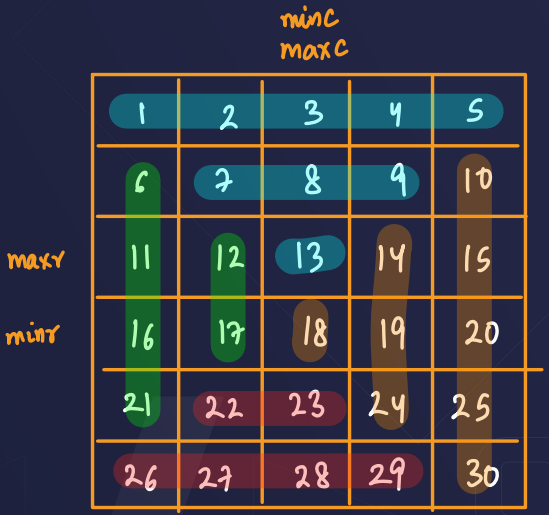
Q9 : Write a program to print the matrix in spiral form.



Output :

1 2 3 6 9 8 7 4 5

[Leetcode 54]



minr, maxr, minc, maxc
0 m-1 0 n-1

Repeat

Blue
i : minr
j : minc → maxc
minr ++

Orange
i : minr → maxr
j : maxc
maxc--

Red
i : maxr
j : maxc → minc
maxr--

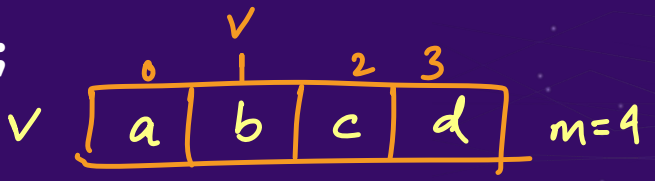
Green
i : maxr → minr
j : minc
minc++

2D ArrayList

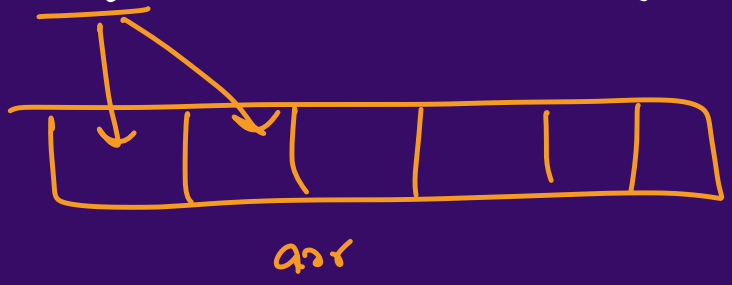
- `List<List<Integer>> v = new ArrayList<>();`
- `List<List<Integer>> v = new ArrayList<>(m);`

```
List<List<Integer>> v = new ArrayList<>(m);
for (int i = 0; i < m; i++) {
    v.add(new ArrayList<>(n));
}
```

Creates empty list



List < Integer > arr = new ArrayList<>(); → 1D arraylist



Advantages of ArrayList over arrays

PW SKILLS

1) Variable size

2) The ArrayLists inside the ArrayList can be of different sizes

a = {10, 20, 30}

b = {40, 50}

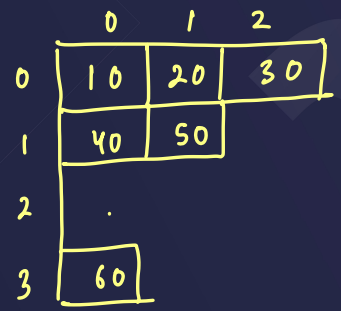
c = {60, 70, 80, 90, 100}

V = {a, b, c}

Basic STL functions in ArrayList

- add() → add at back
- remove() → remove from back
- get()
- size()
- clear()
- isEmpty()
- Contains()
- indexOf()
- toArray()
  ↓
  1D ArrayList to Array.

```
List<Integer> a = new ArrayList<>();
a.add(10); a.add(20); a.add(30);
List<Integer> b = new ArrayList<>();
b.add(40); b.add(50);
List<Integer> c = new ArrayList<>();
List<Integer> d = new ArrayList<>();
d.add(60);
List<List<Integer>> l = new ArrayList<>();
l.add(a); l.add(b); l.add(c); l.add(d);
```



a = { 10, 20, 30 } (0 1 2)

b = { 40, 50 } (0 1)

c = { }

d = { 60 } (0)

l = { { 10, 20, 30 }, { 40, 50 }, { }, { 60 } } (0 1 2 3)

Ques:

n = 5

Q10 : Given an integer 'numRows', generate Pascal's triangle.

l.get(i).get(j) = l.get(i-1).get(j)
+ l.get(i-1).get(j-1)

List < List < Integer >> l =

[Leetcode 118]

Ques:

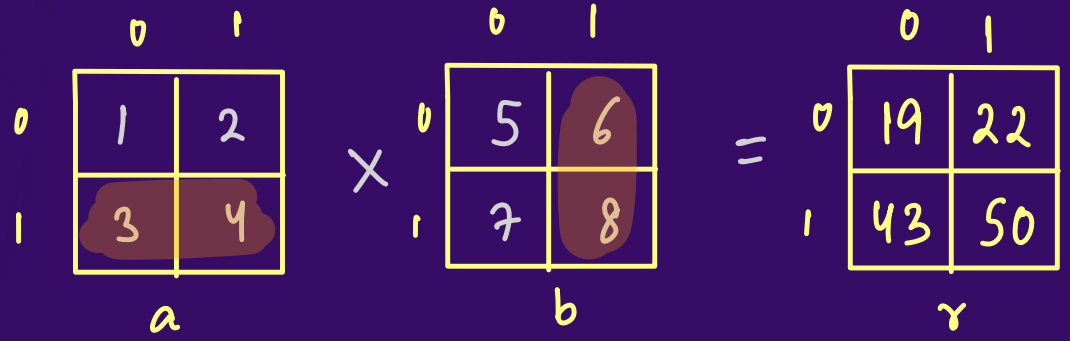
Q11 : Write a program to print the multiplication of two matrices given by the user.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 5 & 12 \\ 21 & 32 \end{bmatrix}$$ wrong ✗

wrong

Ques:

Q11 : Write a program to print the multiplication of two matrices given by the user.



1x5 + 2x7

1x6 + 2x8

15 + 28 = 43

18 + 32

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

X

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

=

	0	1	2
0	30	24	18
1	84	75	
2			

Is square matrix a compulsion ?

	0	1	2
0	1	2	3
1	1	2	3

a — 2×3

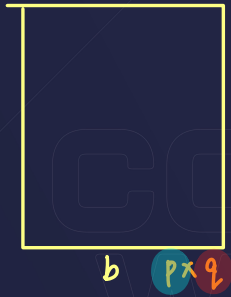
×

	0	1	2
0	4	5	6
1	4	5	6

b — 2×3

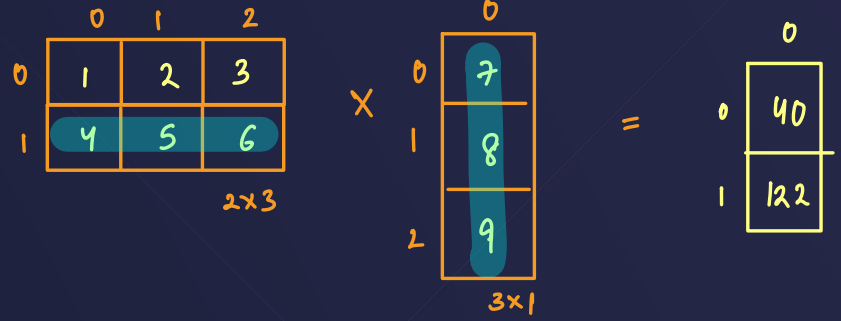
= not possible

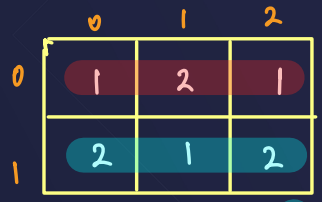
×

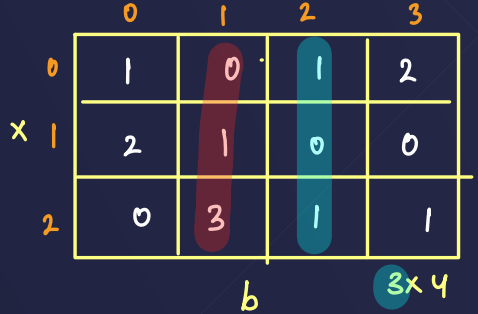


=

only if n == p


0 1 2
0 1 2 3
1 4 5 6
2x3
X
0
0 7
1 8
2 9
3x1
=
0
0 40
1 122

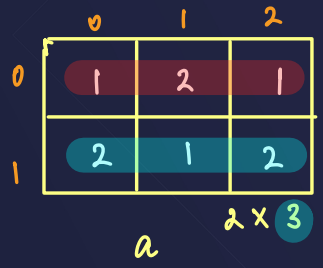


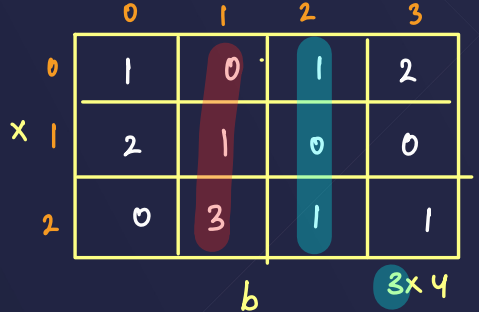


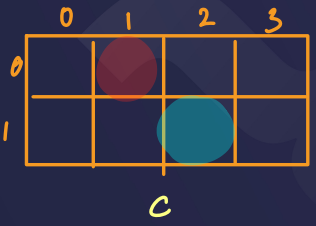

```
int[][] c = new int[a.length][b[0].length];

for(int i=0; i<c.length; i++){
    for(int j=0; j<c[0].length; j++){
        |
        |
        3
    |
    3
```

Logic ?

```
for(int k=0; k<b.length; k++)
    |
    |
    3
```







$$c[0][1] = a[0][0] \times b[0][1] + a[0][1] \times b[1][1] + a[0][2] \times b[2][1]$$

$$c[i][j] = a[i][0] \times b[0][j] + a[i][1] \times b[1][j] + a[i][2] \times b[2][j];$$

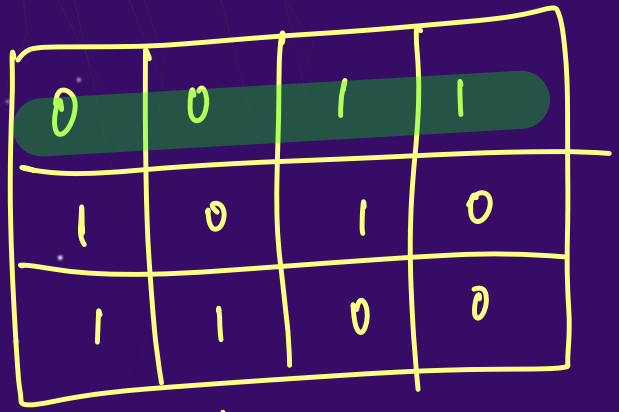
$$c[i][j] = \sum_{r=0}^{b.len-1} a[i][r] \times b[r][j]$$

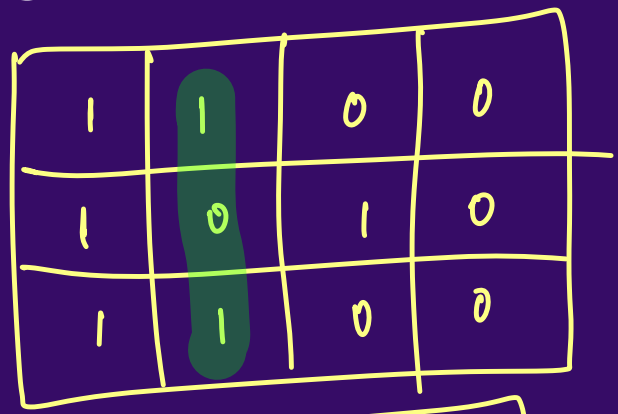
Ques:

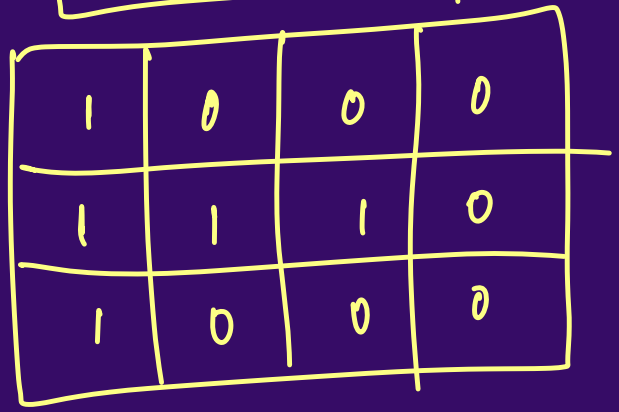
1 10 11 100

PW SKILLS

Q12 : Score after flipping matrix .







↓

3
+
10 = 25
+
12

[Leetcode 861]

Ques:

PW SKILLS

Q12 : Score after flipping matrix .

Hint :

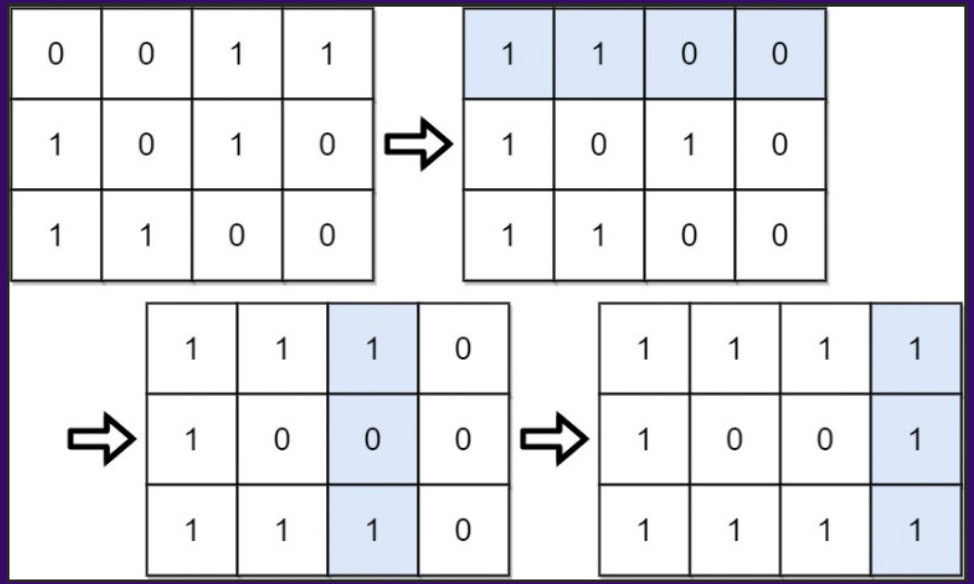
1) 100100 > 001111

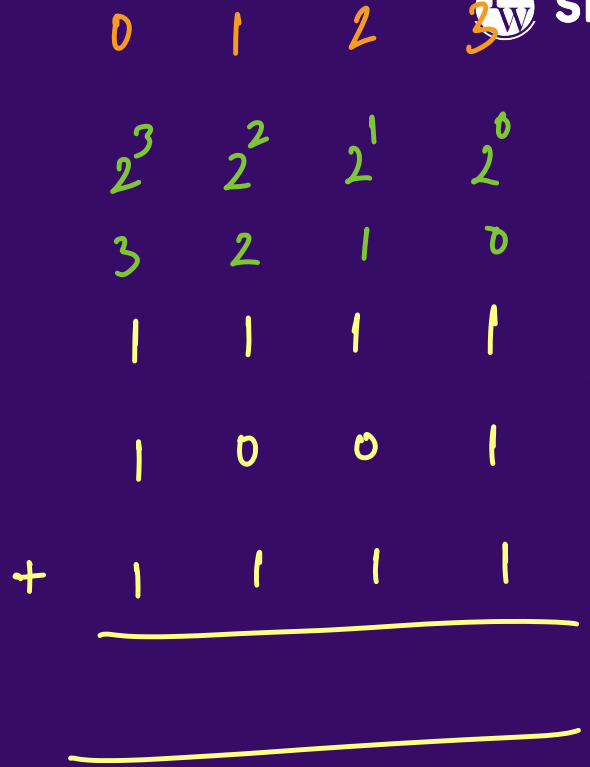
2) 1110 > 1101

[Leetcode 861]

Ques:

Q12 : Score after flipping matrix .





[Leetcode 861]

Ques:

PW SKILLS

Q13 : Write an efficient algorithm that searches for a value target in an m x n integer matrix which has the following properties :

- **Integers in each row are sorted in ascending from left to right.**
- **Integers in each column are sorted in ascending from top to bottom.**

[Leetcode 240]

i

1	4	7	11	15
2	5	8	12	19
3	6	9	16	22
10	13	14	17	24
18	21	23	26	30

j

target = 20

Left, Down

1	4	7	11	15
2	5	8	12	19
3	6	9	16	22
10	13	14	17	24
18	21	23	26	30

t = 16

1	4	7	11	15
2	5	8	12	19
3	7	9	16	22
10	13	14	17	24
18	21	23	26	30

t = 6

Ques:

PW SKILLS

Q14 : Given an m x n integer matrix matrix, if an element is 0, set its entire row and column to 0's.

[Leetcode 73]

Method-1 Using a copy of original array to traverse & check for 0's (m*n space)

1	1	1
1	0	1
1	1	1

arr

1	1	1
1	0	1
1	1	1

helper

We are using m*n 'int's

Method-2 : Thoda sa aur optimised (Using less space) (m+n) Space

	0	1	2	3
0	0	1	2	0
1	3	4	5	2
2	1	3	1	5

arr

```
boolean[] row = new boolean[m];
```


```
boolean[] col = new boolean[n];
```

Method - 3 : Using constant extra space

	0	1	2	3	4
0	1	2	2	11	21
1	3	0	24	21	9
2	6	10	7	0	-8
3	8	4	19	3	99

arr

	0	1	2	3	4
0	1	0	2	0	21
1	0	0	0	0	0
2	0	0	0	0	0
3	8	0	19	0	99

arr

	0	1	2	3	4
0	1	2	2	0	21
1	3	0	24	21	9
2	6	10	7	0	-8
3	0	4	19	3	99

arr

VVVIMP :

```
boolean zeroRow = false;
boolean zeroCol = false;
```

	0	1	2	3	4
0	1	0	2	0	21
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

arr

zeroRow = true

zeroCol = true

THANK YOU