2D Arrays in One Shot

What and Why?

So far we have explored arrays with only one dimension. It is also possible for arrays to have two or more dimensions. The two dimensional array is also called a matrix.

datatype array_name[r][c];

This is a 2D array where r depicts number of rows in matrix and c depicts number of columns in the matrix.

maths marks /10

arr

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

arr[3] = 8

3 → index

		0	1	2	3	4	5	6
Phy	0	1	2	3	4	5	6	35
Chem	1	7	8	9	10	~~11~~ 100	1	2
Math	2	3	4	5	11	13	12	15

15 → cell

brr

brr[1][4] = 100

brr[r][c]

r → row no.

c → column number

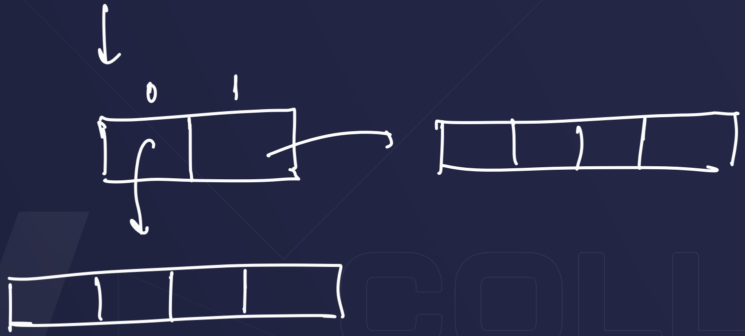
```
int arr[3][2];
```

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

```
arr[0][0] = 1;
arr[0][1] = 2;
   .
   .
   .
   .
```

* What is actually 2D array

int arr [2] [4]



How to print the elements

int a [2][3] ;

	0	1	2
0	a[0][0]	a[0][1]	a[0][2]
1	a[1][0]	a[1][1]	a[1][2]

	0	1	2
0	10	9	8
1	7	6	5

```
                   2
for (int i = 0 ; i < r ; i++){
                          3
     for (int j = 0 ; j < c ; j++){
          printf("%d", a[i][j]);
     }
     printf("\n");
}
```

Initialisation of a 2-Dimensional Array

```
int arr[4][2] = { { 1234, 56 }, { 1256, 43 }, { 1434, 32 }, { 1312, 96 } } ;

int arr[4][2] = { 1234, 56 , 1256, 43 , 1434, 32 , 1312, 96} ;

int arr[2][3] = { 12, 34, 56, 78, 91, 23 } ;

int arr[ ][3] = {12, 34, 56, 78, 91, 23  } ;
```

```
int a[2][2] = { {1,2} , {3, 4} } / {1, 2, 3, 4}
```

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

		R.No	Marks
		0	1
Raghav	0	76	80
Sanket	1	57	81
Urvi	2	40	90
Manvi	3	21	95

#H.W: User input → no. of students

Marks of P, C, M

```
int arr[4][2] = {76,80, 57,81, 40, 90, 21, 95};
```

Ques : Write a program to store 10 at every index of a 2D matrix with 5 rows and 5 columns.

	0	1	2	3	4
0	10	10	10	10	10
1	10	10	10	10	10
2	10	10	10	10	10
3	10	10	10	10	10
4	10	10	10	10	10

```
int arr[5][5] = {10,10,10, . . . . }
```

Ques : Write a program to add two matrices.

$$\begin{array}{c|c|c|} & 0 & 1 \\ \hline 0 & 1 & 2 \\ \hline 1 & 3 & 4 \\ \hline \end{array} + \begin{array}{c|c|c|} & 0 & 1 \\ \hline 0 & 5 & 6 \\ \hline 1 & 7 & 8 \\ \hline \end{array} \qquad \begin{array}{c|c|c|} & 0 & 1 \\ \hline 0 & 6 & 8 \\ \hline 1 & 10 & 12 \\ \hline \end{array}$$

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

int res[2][2];
```

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

H.W : Do it without using extra matrix

Ques : Find the sum of a given matrix of n x m.

rows

columns

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

Sum = 0;

Homework : 1) Find out the max^m element & min^m element in a 2D-array

2) & the index of max^m element → (i, j)

HW : Given a matrix ‘a’ of dimension n x m and 2 coordinates (l1, r1) and (l2, r2). Return the sum of the rectangle from (l1,r1) to (l2, r2).

int a[m][n];

a[i][j] → (i,j)

	0	1	2	3	4
0	(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
1	(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
2	(2,0)	(2,1)	(2,2)	(2,3)	(2,4)
3	(3,0)	(3,1)	(3,2)	(3,3)	(3,4)

Homework : Write a program to print the row number having the maximum sum in a given matrix. & the maxSumRow

$$
\begin{array}{c|cccc|c}
 & 0 & 1 & 2 & 3 & \\
\hline
0 & 1 & 2 & 3 & 1 & 7 \\
1 & 0 & 5 & 0 & 2 & 7 \\
2 & 9 & 0 & 0 & 3 & 12 \\
\hline
\end{array}
$$

maxSum =

Ques : Given a matrix having 0-1 only, find the row with the maximum number of 1's.

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

Ques : Write a program to Print the transpose of the matrix entered by the user. **(Leetcode - 867)**

transpose ?

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 2 & 3 \\ 1 & 4 & 5 & 6 \end{array} \rightarrow \begin{array}{cc} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{array}$$

arr[2][3]

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & (0,0) & (0,1) & (0,2) \\ 1 & (1,0) & (1,1) & (1,2) \end{array} \qquad \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & (0,0) & (1,0) \\ 1 & (0,1) & (1,1) \\ 2 & (0,2) & (1,2) \end{array}$$

Ques : Write a program to Print the transpose of the matrix entered by the user. (Leetcode - 867)

& store it in a separate matrix

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 2 & 3 \\ 1 & 4 & 5 & 6 \end{array} \rightarrow \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 1 & 4 \\ 1 & 2 & 5 \\ 2 & 3 & 6 \end{array}$$

arr[2][3]

brr[3][2]

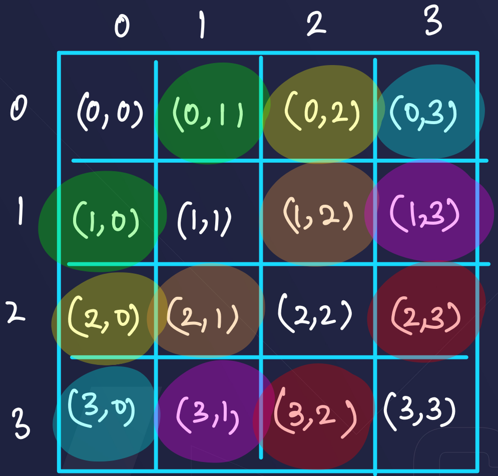
brr[i][j] = arr[j][i];

* Ques : Write a program to ~~Print the transpose of the matrix entered by the user.~~ (Leetcode - 867)

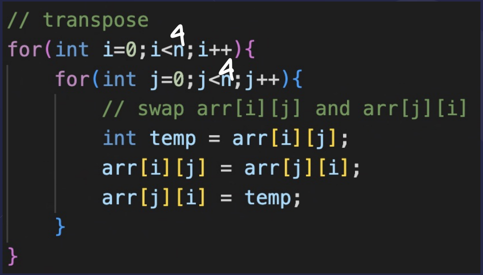
to change the given $\boxed{n \times n}$ matrix to its transpose.

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 2 & 3 \\ 1 & 4 & 5 & 6 \\ 2 & 7 & 8 & 9 \end{array} \quad \longrightarrow \quad \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 4 & 7 \\ 1 & 2 & 5 & 8 \\ 2 & 3 & 6 & 9 \end{array}$$

arr[n][n]; arr[n][n];

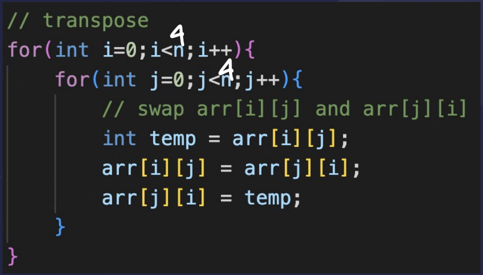


	0	1	2	3
0	(0,0)	(0,1)	(0,2)	(0,3)
1	(1,0)	(1,1)	(1,2)	(1,3)
2	(2,0)	(2,1)	(2,2)	(2,3)
3	(3,0)	(3,1)	(3,2)	(3,3)



```
// transpose
                 4
for(int i=0;i<n;i++){
                    4
    for(int j=0;j<n;j++){
        // swap arr[i][j] and arr[j][i]
        int temp = arr[i][j];
        arr[i][j] = arr[j][i];
        arr[j][i] = temp;
    }
}
```

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



```
// transpose
              4
for(int i=0;i<n;i++){
                  4
    for(int j=0;j<n;j++){
        // swap arr[i][j] and arr[j][i]
        int temp = arr[i][j];
        arr[i][j] = arr[j][i];
        arr[j][i] = temp;
    }
}
```

i → 0 to n-1

j → $\boxed{i \text{ to } n-1}$

or

$\boxed{0 \text{ to } i}$

Ques : Write a program to rotate a matrix 90° clockwise. (Leetcode - 48)

original

1	2	3
4	5	6
7	8	9

arr[3][3]

transpose

1	4	7
2	5	8
3	6	9

rotated 90°

7	4	1
8	5	2
9	6	3

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

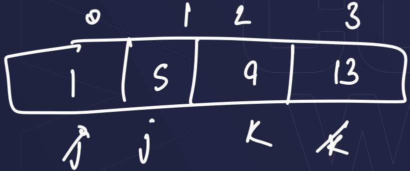
1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

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

Steps :

1) Transpose

2) Reverse each row



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

$$\begin{array}{c@{}c} & \begin{array}{cc} 0 & 1 \end{array} \\ \begin{array}{c} 0 \\ 1 \end{array} & \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \end{array} \times \begin{array}{c@{}c} & \begin{array}{cc} 0 & 1 \end{array} \\ \begin{array}{c} 0 \\ 1 \end{array} & \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \end{array} = \begin{array}{c@{}c} & \begin{array}{cc} 0 & 1 \end{array} \\ \begin{array}{c} 0 \\ 1 \end{array} & \begin{bmatrix} 1\times5+2\times7 & 1\times6+2\times8 \\ 3\times5+4\times7 & 3\times6+4\times8 \end{bmatrix} \end{array} = \begin{bmatrix} 19 & 22 \\ 36 & 50 \end{bmatrix}$$

a[2][2]

↓

row is dependent on this

b[2][2]

↓

column is dependent on this

res[2][2]

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 2 & 1 \\ 1 & 2 & 1 & 2 \\ 2 & 1 & 2 & 1 \end{array} \times \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 2 & 1 & 2 \\ 1 & 1 & 2 & 1 \\ 2 & 2 & 1 & 2 \end{array} = \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 6 & 6 & 6 \\ 1 & & & \\ 2 & & & \end{array}$$

$$1 \times 2 + 2 \times 1 + 1 \times 2 = 6$$

$$1 \times 1 + 2 \times 2 + 1 \times 1 = 6$$

Rules for matrix multiplication :

$$a[m][n] \times b[p][q] = res[m][q]$$

1) $n == p$

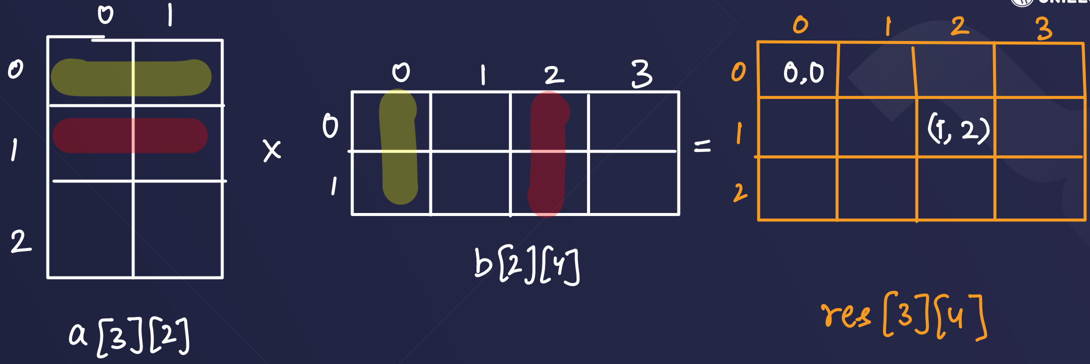
2) resultant order is $m \times q$

3) $A \times B \neq B \times A$

$$\underset{a[2][1]}{\begin{array}{c|c} & 0 \\ \hline 0 & 1 \\ 1 & 2 \end{array}} \times \underset{b[1][2]}{\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 3 & 4 \end{array}} = \underset{res[2][2]}{\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 3 & 4 \\ 1 & 6 & 8 \end{array}}$$

$$\underset{b[1][2]}{\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 3 & 4 \end{array}} \times \underset{a[2][1]}{\begin{array}{c|c} & 0 \\ \hline 0 & 1 \\ 1 & 2 \end{array}} = \underset{res[1][1]}{\begin{array}{c|c} & 0 \\ \hline 0 & 11 \end{array}}$$

$3 \times 1 + 4 \times 2 = 3 + 8 = 11$



res[1][2] = a[1][0] * b[0][2] + a[1][1] * b[1][2];

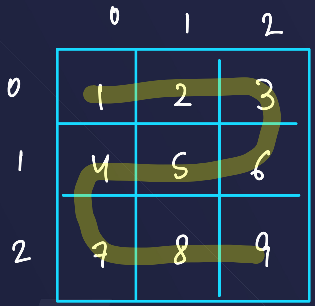
res[i][j] = i^{th} row of a * j^{th} column of b

res[i][j] = (a[i][0], a[i][1], a[i][2]) * (b[0][j], b[1][j], b[2, j])

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}_{3 \times \boxed{2}} \times \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}_{\boxed{2} \times 4} = \begin{bmatrix} 11 & 14 & 17 & 20 \\ 23 & 30 & 37 & 44 \\ 35 & 46 & 57 & 68 \end{bmatrix}_{3 \times 4}$$

$$res[i][j] = \sum_{k=0}^{n} a[i][k] * b[k][j]$$

Q. Wave print - 1



$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 2 & 3 \\ 1 & 4 & 5 & 6 \\ 2 & 7 & 8 & 9 \end{array}$$

Output

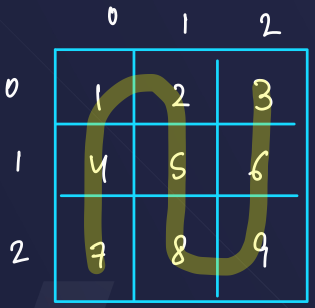
```
1   2   3

6   5   4

7   8   9
```

/ 1 2 3 6 5 4 7 8 9

H.W: Wave print - 2

no of column= n
no of rown = m

a[m][n]



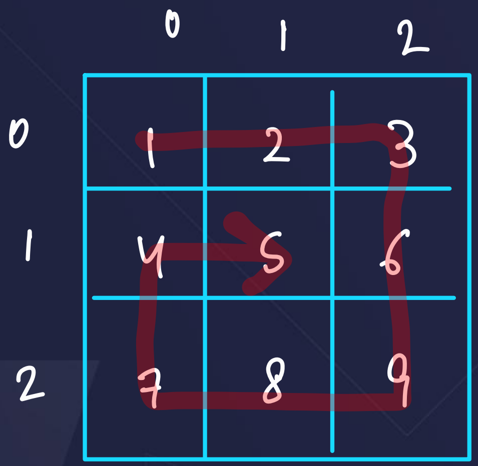
	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

Algo:

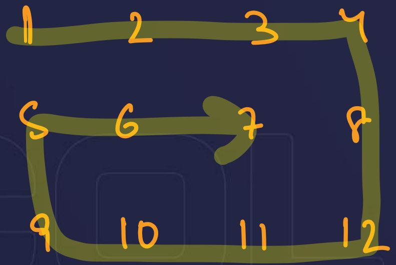
```
if(column no == even){
        rowno → m-1 to 0
}
else{
        rowno → 0 to m-1
}
```

Ques : Given an n x m matrix ‘a’, print all elements of the matrix in spiral order. **(Leetcode - 54)**

Output : 1 2 3 6 9 8 7 4 5

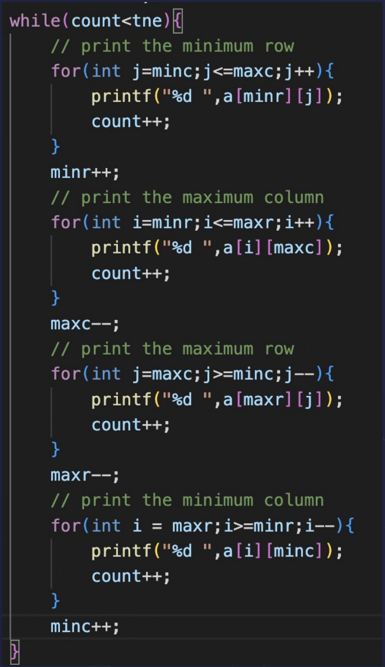


$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 2 & 3 \\ 1 & 4 & 5 & 6 \\ 2 & 7 & 8 & 9 \end{array}$$

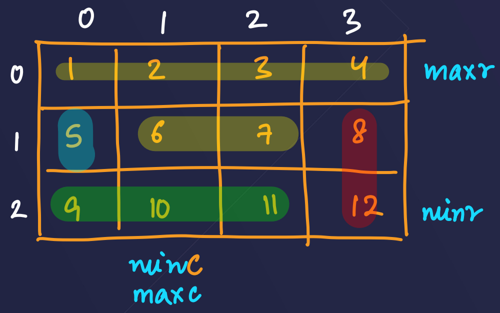


$$\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{array}$$

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



```
while(count<tne){
    // print the minimum row
    for(int j=minc;j<=maxc;j++){
        printf("%d ",a[minr][j]);
        count++;
    }
    minr++;
    // print the maximum column
    for(int i=minr;i<=maxr;i++){
        printf("%d ",a[i][maxc]);
        count++;
    }
    maxc--;
    // print the maximum row
    for(int j=maxc;j>=minc;j--){
        printf("%d ",a[maxr][j]);
        count++;
    }
    maxr--;
    // print the minimum column
    for(int i = maxr;i>=minr;i--){
        printf("%d ",a[i][minc]);
        count++;
    }
    minc++;
}
```



$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 1 & 2 & 3 & 4 \\ 1 & 5 & 6 & 7 & 8 \\ 2 & 9 & 10 & 11 & 12 \end{array}$$

r = 3

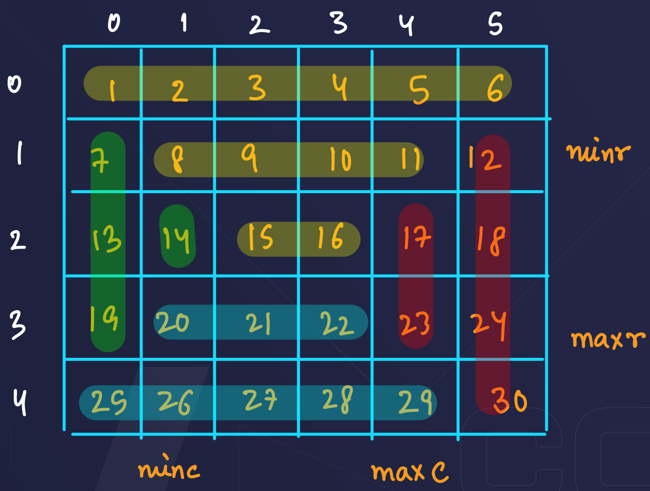
c = 4

tne = 12

Output

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

Count = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~ ~~10~~ ~~11~~ ~~12~~ 13



	0	1	2	3	4	5
0	1	2	3	4	5	6
1	7	8	9	10	11	12
2	13	14	15	16	17	18
3	19	20	21	22	23	24
4	25	26	27	28	29	30

```
minr ✓
minr++  →  loop{
               a[minr][col]
           }  col → minc to maxc

maxc
maxc--   (minr → maxr)

maxr [reverse]
maxr--   maxc → minc

minc [reverse]
minc++   maxr → minr
```

$a[5][6]$

`int tne = m*n;`   `count < tne`

HW : Given a positive integer n, generate a n x n square matrix filled with elements from 1 to n^2 in spiral order. (Leetcode - 59)

$n = 3$

1	2	3
8	9	4
7	6	5

a[i][j]