Ques. Given A and B matrices. Write a function that returns a matrix having A+B.

A =

	0	1	2	3
0	0	-1	5	4
1	2	9	8	15
2	7	6	-5	8

3 x 4

B =

	0	1	2	3
0	9	5	6	11
1	3	-2	7	-10
2	4	3	1	0

3 x 4

ans =

	0	1	2	3
0	9	4	11	15
1	.	7	15	
2	11			

ans [1][2] = 9 + (-2) = 7

= A[1][2] + B[1][2]

Dimensions of A and B should be same

Dimension of output matrix will also be same as A and B.

ans [1][2] = A[1][2] + B[1][2]

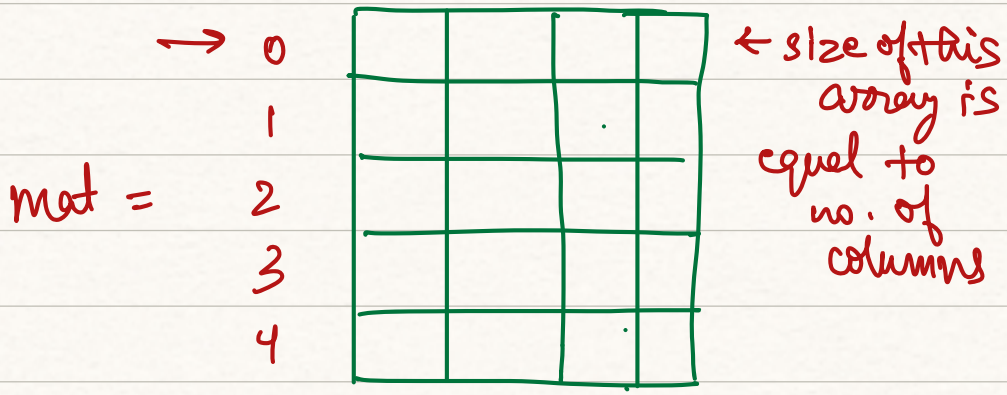
= 8 + 7 = 15

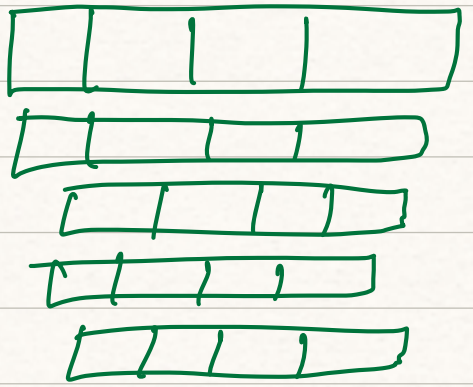
ans[i][j] = A[i][j] + B[i][j]

```
int [][] sumOfMatrices (int [][] A, int [][] B) {
    int [][] ans = new int [A.length] [A[0].length];
    for (int i = 0 ; i < A.length; i++) {
        for (int j = 0; j < A[0].length; j++) {
            ans [i][j] = A[i][j] + B [i][j];
        }
    }
    return ans;
}
```

i = 2 j = 0

ans [2][0] = A[2][0] + B[2][0]
= 7 + 4 = 11





no. of rows = mat.length

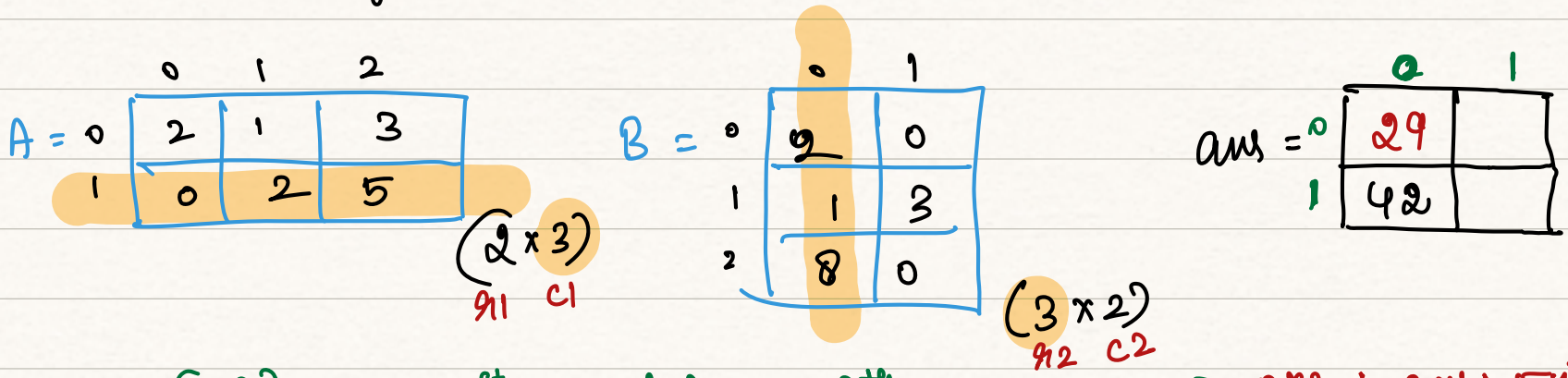
no. of columns = mat[0].length → 4

mat[2].length → 4

mat[4].length → 4

Matrix Multiplication

Ques. Write a function that returns a matrix having A x B.



(1, 0) → 1st row of A * 0th column of B ⇒ 0x2 + 2x1 + 5x8

Rule 1 → the number of columns in A should be equal to number of row B. [c1 == r2]

Rule 2 → Dimension of output matrix will be (r1, c2)

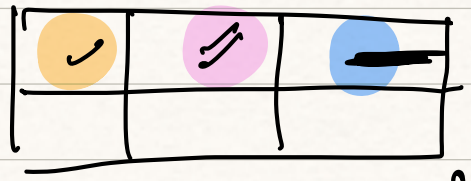
ans [0] [0] - (0th row of A) * 0th colum of B

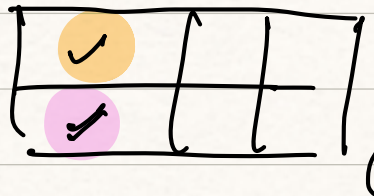
$2*2 + 1*1 + 3*8$

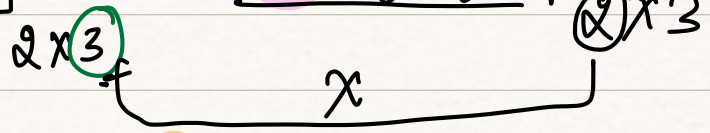
$\Rightarrow 4 + 1 + 24$

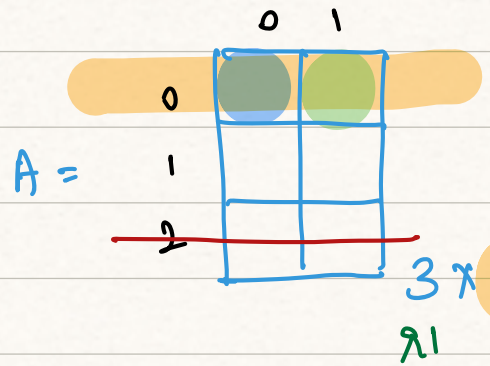
$\Rightarrow 29$

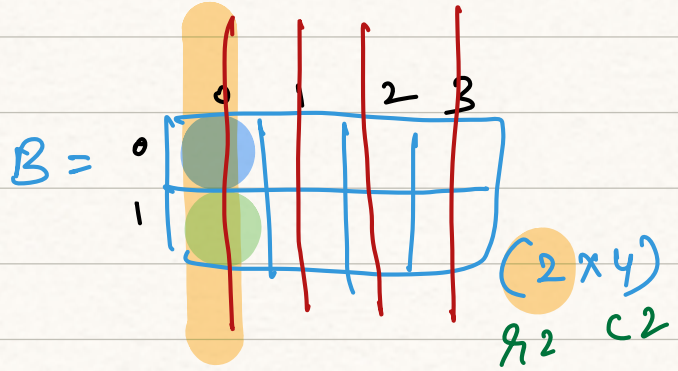
(Not possible)







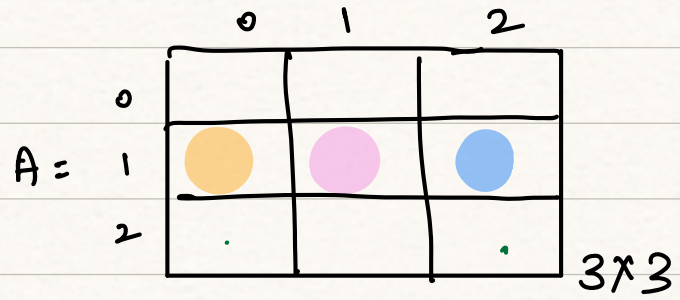


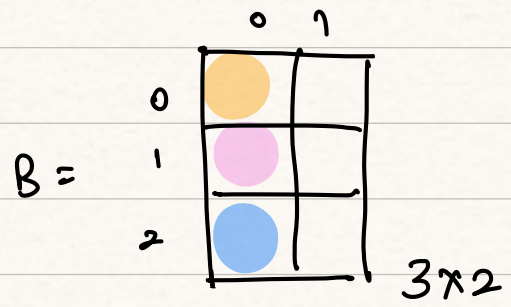


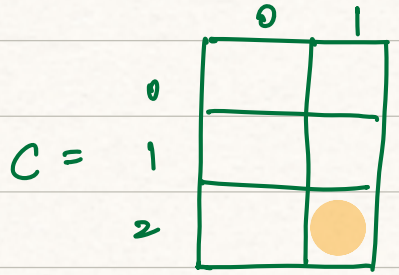
ans =

$$\begin{bmatrix} (0,0) & (0,1) & (0,2) & (0,3) \\ (1,0) & (1,1) & (1,2) & (1,3) \\ (2,0) & (2,1) & (2,2) & (2,3) \end{bmatrix}$$

3 x 4
r1, c2







$C[2][1] \Rightarrow$ (2nd row of A) * (1st column of B)

$C[2][1] = A[2][0] * B[0][1] + A[2][1] * B[1][1] + A[2][2] * B[2][1]$

$C[1][0] \Rightarrow$ (1st row of A) * (0th column of B)

$A[1][0] * B[0][0] + A[1][1] * B[1][0] + A[1][2] * B[2][0]$

$C[i][j] \rightarrow$ (i^{th} row of A) * (j^{th} column of B)

$A[i][0] * B[0][j] + A[i][1] * B[1][j] + A[i][2] * B[2][j]$

k = 0 , 1 , 2

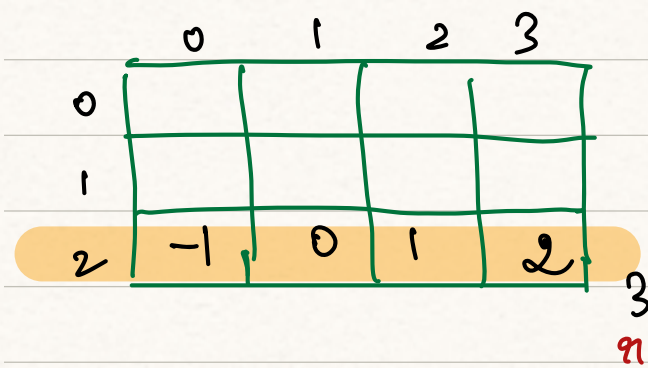
A[i][k] * B[k][j]

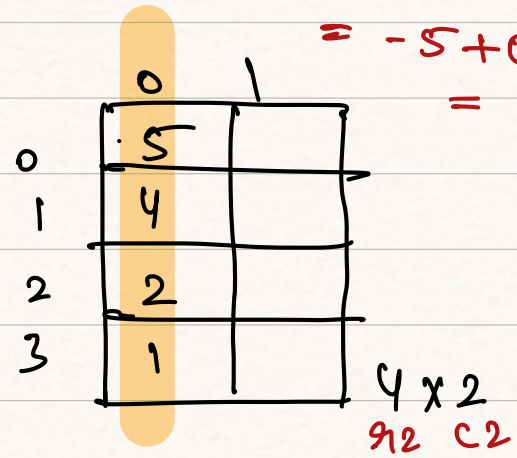
Note → If Rule1 is not followed, then don't call the function

```
int[][] matrix_mul (int[][] A, int[][] B) {

    int r1 = A.length;
    int c1 = A[0].length;
    int r2 = B.length;
    int c2 = B[0].length;

    int[][] ans = new int[r1][c2];
// we have to fill the ans matrix at every cell
```

c1 == r2

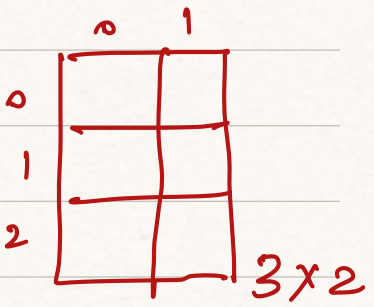
```
for (int i = 0 ; i < r1 ; i++) {
    for (int j = 0 ; j < c2 ; j++) {

        int temp = 0 ;

        for (int k = 0 ; k < c1 ; k++) {

            temp = temp + A[i][k] * B[k][j];

        }

        ans [i] [j] = temp;

    }
}
```





$-1 \times 5 + 0 \times 4 + 1 \times 2 + 2 \times 1$
$= -5 + 0 + 2 + 2$
$= -1$

ans =



i = 2 j = 0

ans [2][0]

```
int temp = 0;

for ( int k = 0; k < c1; k++) {

    temp = temp + A[i][k] * B[k][j];

}

ans[i][j] = temp;
```

temp = 0 i = 2 j = 0 ans[2][0] = -1

k	k<4	temp
0	true	0 + A[2][0] * B[0][0] = -5
		-1 * 5
1	true	-5 + A[2][1] * B[1][0] = -5
		0 * 4

2 true $-5 + A[2][2] * B[2][0] = -3$

$1 * 2$

3 true $-3 + A[2][3] * B[3][0] = -1$

$2 * 1$

4 false ⟶ break

Break till 10:35 pm

A (3 x 4)
r1 c1

B (4 x 7)
r2 c2

c1 == r2 ⇒ true

A x B ⇒ Yes

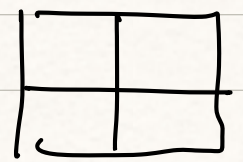
A (2 x 2)
r1 c1

B (2 x 2)
r2 c2

c1 == r2 ⇒ true

A x B ⇒ Yes

Square



2 x 2

A (3 x 7)
r1 c1

B (4 x 7)
r2 c2

A x B ⇒ No

$A = (3 \times 4)$ $\quad$ $B = (4 \times 7)$

$A \times B \Rightarrow$ Yes

$B \times A \Rightarrow$ NO

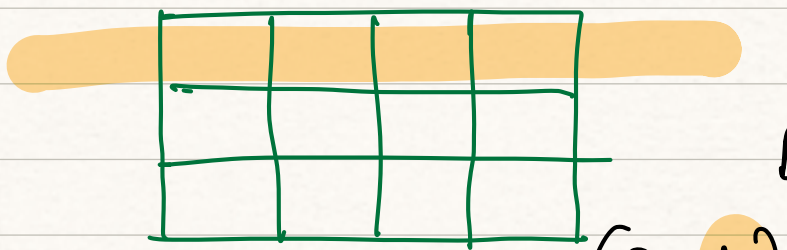
$B = (4 \times 7)$ $\quad$ $A = (3 \times 4)$

$r_1 \; c_1$ $\quad$ $r_2 \; c_2$

$A = (3, 4)$ $\quad$ $B = (5, 3)$

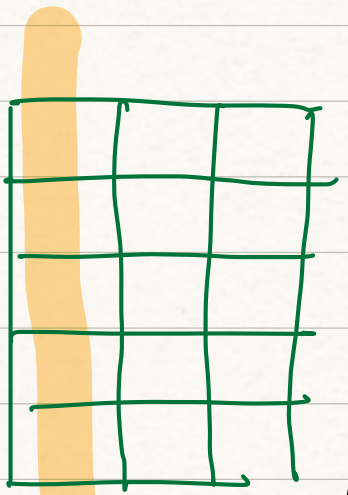
$r_1 \; c_1$ $\quad$ $r_2 \; c_2$

A



(3×4)

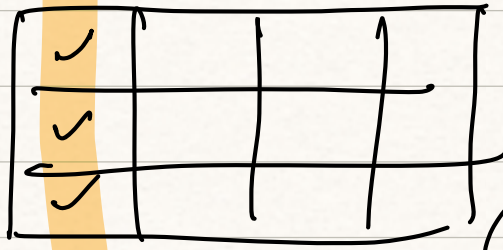
$B =$



(5×3)

B = 

(5 x 3)

A = 

(3 x 4)

Transpose of a matrix

Ques Given a matrix, you have return the transpose

A =

	0	1
0	10	20
1	30	40
2	50	60
3	70	80

(4 x 2)
N x M

A^T =

	0	1	2	3
0	10	30	50	70
1	20	40	60	80

(2 x 4)
M x N

(0, 3) ⇒ 70 (3, 0)

A^T (i, j) A(j, i)

element	A^T	A
10	0,0	0,0
20	1,0	0,1
50	0,2	2,0
80	1,3	3,1
	i , j	j , i

$$A^T[i][j] = A[j][i];$$

```
int[][] transpose (int[][] A) {
    int N = A.length;
    int M = A[0].length;
    int[][] ans = new int[M][N];
```

```
for (int i=0 ; i < M ; i++) {
    for ( int j=0; j<N; j++) {
        ans [i] [j] = A [j] [i];
    }
}
```

N = 2 , M = 3

A =

	0	1	2
0	1	2	3
1	4	5	6

(2, 3)

ans =

	0	1
0	1	4
1	2	5
2	3	6

(3 x 2)

i	i < 3	j	j < 2
0	✓	0	✓
		1	✓
		2	x break
1	✓	0	✓
		1	✓
		2	x break
2	✓	0	✓
		1	✓
		2	x break
3	x break		

(0,0) => (0,0)
(0,1) => (1,0)
(1,0) => (0,1)
(1,1) => (1,1)
(2,0) => (0,2)
(2,1) => (1,2)

Identity Matrix

square matrix

① [no. of rows = no. of columns]

[1 x 1] [2 x 2] [3 x 3] [4 x 4]

② only diagonal element are 1. rest elements are 0.

1	0
0	1

2 x 2

1	0	0
0	1	0
0	0	1

3 x 3

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

4 x 4

(3) $A * I = A$

1	2	3
4	5	6

2x3

$*$

1	0	0
0	1	0
0	0	1

3x3

$=$

1	2	3
4	5	6

2x3

$1 \times 1 + 2 \times 0 + 3 \times 0$

$1 \times 0 + 2 \times 1 + 3 \times 0$

$1 \times 0 + 2 \times 0 + 3 \times 1$

$4 \times 1 + 5 \times 0 + 6 \times 0$

$4 \times 0 + 5 \times 1 + 6 \times 0$

$4 \times 0 + 5 \times 0 + 6 \times 1$

ans =

1	2	3
4	5	6

2x3

NxM → dimension of Identity matrix

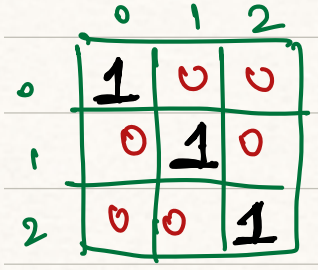
MxM

creating identity matrix for a given matrix A.

```
int[][] identity (int[][] A){
    int M = A[0].length;
    int[][] ans = new int[M][M];
    for(int i=0; i<M; i++){
        ans[i][i] = 1;
    }
    return ans;
}
```

i ==> M=3

M=3



0,0 ✓
1,1 ✓
2,2 ✓

i = 0 ⟶ ans[0][0]
i = 1 ⟶ ans[1][1]
i = 2 ⟶ ans[2][2]

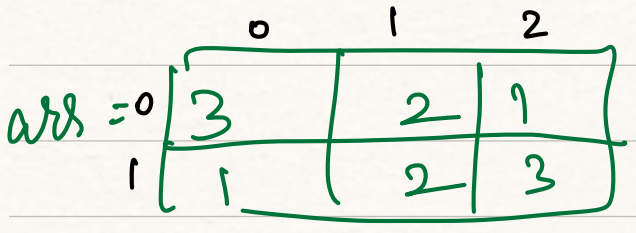
$$A = \begin{matrix} 1 & 4 & 6 \\ 5 & 7 & 2 \\ 8 & 9 & 5 \end{matrix} \qquad A^T = \begin{matrix} 1 & 5 & 8 \\ 4 & 7 & 9 \\ 6 & 2 & 5 \end{matrix}$$

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

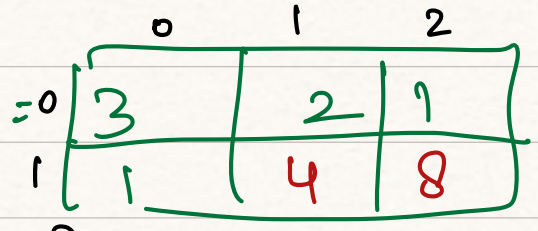
i = 0 [0][1]
1 [1][1]
2 [2][1]
3 [3][1]

```
for(int i = 0;   i < 4 ; i++)

    matrix[i][1]
```

2 5 9 13



(2 x 3)



(1,2)

```
for (row=1 ; row<2 ; row++){
    for(col=1; col<3; col++){
        if (arr[row][col] %2==1){
            +1;
        }
        if (arr[row][col] %2==0){
            *2;
        }
    }
}
```

row	row<2
1	✓

col	col<3
1	✓
2	✓