2D Array

(a) Find the summation of the elements of a 2D array.

	0	1	2
0	12	5	8
1	6	7	4
2	18	9	2

sum	71

Input (*Declarations and Initializations***):** int arr[3][3], int sum = 0;

Process:

1. Start from the row with ***row-value*** r=0.
2. Start from the column with ***column-value*** c= 0.
3. Add the value of **sum** with the **element** in **arr[r][c]** index.
4. Store the summation of the add operation in (3) in **sum**.
5. Increase the value of **c** by 1.
6. Repeat (3), (4) and (5) for all the columns.
7. Increase the value of **r** by 1.
8. Repeat (2), (3), (4), (5), (6), (7) for all the rows.

Output: Print the value of **sum.**

(b) Find the summation of two 2D arrays and store the result in another 2D array.

	0	1	2
0	12	5	8
1	6	7	4
2	18	9	2

+

	0	1	2
0	2	8	18
1	16	14	5
2	8	19	12

=

	0	1	2
0	14	13	26
1	22	21	9
2	26	28	24

Input (*Declarations and Initializations***):** int A[3][3], int B[3][3], int S[3][3];

Process:

1. Start from the row with ***row-value*** r=0.
2. Start from the column with ***column-value*** c= 0.
3. Add the value of **A[r][c]** with **B[r][c]** and store the summation in **S[r][c]**.
4. Increase the value of **c** by 1.
5. Repeat (3) and (4) for all the columns.
6. Increase the value of **r** by 1.
7. Repeat (2), (3), (4), (5) and (6) for all the rows.

Output: Print the array **S.**

(c) Find the Summation of the boundary elements of a 2D array.

	0	1	2	3
0	12	5	8	10
1	6	7	4	11
2	18	9	2	1
3	20	3	15	13

Input (*Declarations and Initializations***):** int arr[4][4], int sum = 0;

Process:

1. Start from the row with ***row-value*** r=0.
2. Start from the column with ***column-value*** c= 0.
3. If the value of r is 0 or r is (4-1) or c is 0 or c is (4-1), go to (4), else go to (6).
4. Add the value of **sum** with the **element** in **arr[r][c]** index.
5. Store the summation of add operation in (4) in **sum**.
6. Increase the value of **c** by 1.
7. Repeat (3), (4), (5) and (6) for all the columns.
8. Increase the value of **r** by 1.
9. Repeat (2), (3), (4), (5), (6), (7) and (8) for all the rows.

Output: Print the value of **sum.**

(d) Find the summation of the diagonal elements of a 2D array.

	0	1	2	3
0	12	5	8	10
1	6	7	4	11
2	18	9	2	1
3	20	3	15	13

	0	1	2	3	4
0	12	5	8	10	18
1	6	7	4	11	21
2	18	9	2	1	31
3	20	3	15	13	28
4	30	3	35	23	29

Input (*Declarations and Initializations***):** int arr[n][n], int sum = 0;

Process:

1. Start from the row with ***row-value*** r=0.
2. Start from the column with ***column-value*** c= 0.
3. If the value of **r** is equal to **c** or **(r+c)** is equal to **(n-1)**, go to (4), else go to (6).
4. Add the value of **sum** with the **element** in **arr[r][c]** index.
5. Store the summation of add operation in (4) in **sum**.
6. Increase the value of **c** by 1.
7. Repeat (3), (4), (5) and (6) for all the columns.
8. Increase the value of **r** by 1.
9. Repeat (2), (3), (4), (5), (6), (7) and (8) for all the rows.

Output: Print the value of **sum.**

(e) Find the transpose matrix of a 2D Array.

	0	1	2	3
0	12	5	8	10
1	6	7	4	11
2	18	9	2	1
3	20	3	15	13
4	30	21	35	23

---Transpose--->

	0	1	2	3	4
0	12	6	18	20	30
1	5	7	9	3	21
2	8	4	2	15	35
3	10	11	1	3	23

Input (*Declarations and Initializations***):** int A[5][4], int A_Tr[4][5];

Process:

1. Start from the row with ***row-value*** r=0.
2. Start from the column with ***column-value*** c= 0.
3. Store the element of **A[r][c]** in **A_Tr[c][r].**
4. Increase the value of **c** by 1.
5. Repeat (3) and (4) for all the columns.
6. Increase the value of **r** by 1.
7. Repeat (2), (3), (4), (5) and (6) for all the rows.

Output: Print the array **A_Tr.**