* Matrix Multiplication :

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 2 & 3 \\ 1 & 4 & 5 & 6 \end{array} \qquad \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 10 & 11 \\ 1 & 20 & 21 \\ 2 & 30 & 31 \end{array} \qquad =$$

2×3 $\qquad$ 3×2

	0	1
0	1x10 + 2x20 + 3x30 = 140	1x11 + 2x21 + 3x31 = 146
1	4x10 + 5x20 + 6x30 = 320	4x11 + 5x21 + 6x31 = 335

2×2

* you can only multiply $m_1 \times n_1$ matrix with $m_2 \times n_2$ matrix if and only if $n_1 == m_2$ i.e; no. of cols in mat1 = no. of rows in mat2

* The resultant matrix will be $m_1 \times n_2$

$$2 \times 3 \; * \; 3 \times 2 = 2 \times 2$$

$$(m_1 \times n_1) \; * \; (m_2 \times n_2) = m_1 \times n_2$$

$$
\begin{array}{c|ccc}
 & 0 & 1 & 2 \\
\hline
0 & a_{00} & a_{01} & a_{02} \\
1 & a_{10} & a_{11} & a_{12}
\end{array}
\qquad
\begin{array}{c|cccc}
 & 0 & 1 & 2 & 3 \\
\hline
0 & b_{00} & b_{01} & b_{02} & b_{03} \\
1 & b_{10} & b_{11} & b_{12} & b_{13} \\
2 & b_{20} & b_{21} & b_{22} & b_{23}
\end{array}
=
\begin{array}{c|cccc}
 & 0 & 1 & 2 & 3 \\
\hline
0 & c_{00} & c_{01} & c_{02} & c_{03} \\
1 & c_{10} & c_{11} & c_{12} & c_{13}
\end{array}
$$

2×3 ($R1$) $\qquad 3 \times 4$ ($C2$) $\qquad 2 \times 4$

$R_1 \times C_2$

$$c_{00} = \overbrace{a_{00} \cdot b_{00}}^{0} + \overbrace{a_{01} \cdot b_{10}}^{1} + \overbrace{a_{02} \cdot b_{20}}^{2}$$

$$c_{01} = \overbrace{a_{00} \cdot b_{01}}^{0} + \overbrace{a_{01} \cdot b_{11}}^{1} + \overbrace{a_{02} \cdot b_{21}}^{2}$$

$$c_{02} = \overbrace{a_{00} \cdot b_{02}}^{0} + \overbrace{a_{01} \cdot b_{12}}^{1} + \overbrace{a_{02} \cdot b_{22}}^{2}$$

$$c_{03} = \overbrace{a_{00} \cdot b_{03}}^{0} + \overbrace{a_{01} \cdot b_{13}}^{1} + \overbrace{a_{02} \cdot b_{23}}^{2}$$

c[r][c]

$$= \sum a[r][?] * b[?][c]$$

no. of cols in a / no. of rows in b - 1

$$= \sum_{k=0} a[r][k] * b[k][c]$$

$$C[1][2] = \sum_{k=0}^{2} a[1][k] * b[k][2]$$

$$= a[1][0] * b[0][2] +$$
$$a[1][1] * b[1][2] +$$
$$a[1][2] * b[2][2]$$

* Transpose of a Matrix :

* res[r][c] = mat[c][r]

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

5x5

mat (m x n)

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

5x5 (n x m)

res

⇒ res[0][0] = mat[0][0]

⇒ res[0][1] = mat[1][0]

⇒ res[0][2] = mat[2][0]

⇒ res[0][3] = mat[3][0]

⇒ res[0][4] = mat[4][0]

Interchange rows and cols

1	2
3	4
5	6

3x2

⟺

1	3	5
2	4	6

2x3

```javascript
function matrixTranspose(mat, n) {
  //Write your code here
  const rows1 = n;
  const cols1 = n;

  // r1 x c1 => c1 x r1
  const rows2 = cols1;
  const cols2 = rows1;
  const res = [];

  for (let r = 0; r < rows2; r++) {
    const smallArr = [];
    for (let c = 0; c < cols2; c++) {
      smallArr.push(0);
    }
    res.push(smallArr);
  }

  for (let r = 0; r < rows2; r++) {
    for (let c = 0; c < cols2; c++) {
      res[r][c] = mat[c][r];
    }
  }

  return res;
}
```

new array (extra sp)

TC : $O(n^2 + n^2) = O(2n^2) = O(n^2)$

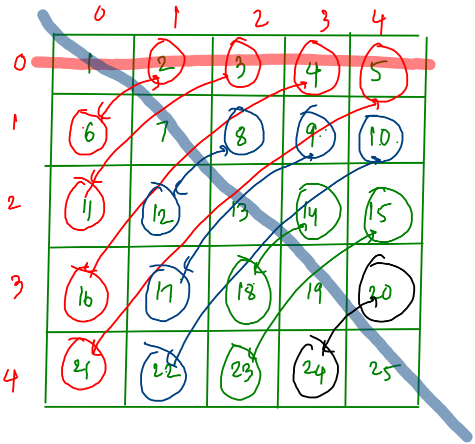
SC : $O(\text{size of ds}) = O(n^2)$

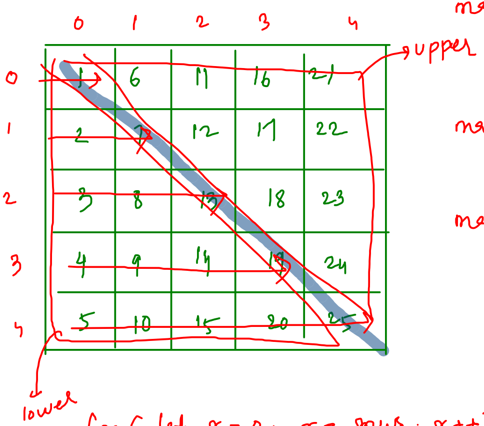
rows2 * cols2 $= n * n = n^2$

+

rows2 * cols2 $= n * n = n^2$

In place solution :





mat[1][0] ⟷ mat[0][1]

mat[3][2] ⟷ mat[2][3]

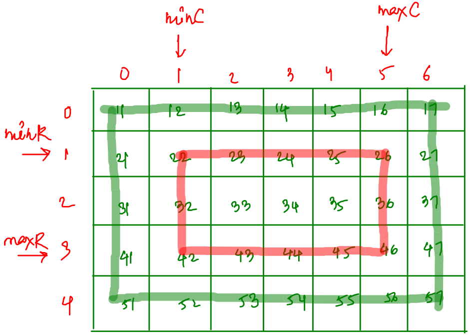
mat[r][c] ⟷ mat[c][r]

```
for (let r = 0; r < rows; r++) {
    for (let c = 0; c < r; c++) {
        swap(m[r][c], m[c][r]);
    }
}
```

* Spiral Traversal :

	0	1	2	3	4	5	6
0	11	12	13	14	15	16	17
1	21	22	23	24	25	26	27
2	31	32	33	34	35	36	37
3	41	42	43	44	45	46	47
4	51	52	53	54	55	56	57

op: 11 12 13 14 15 16 17 27 37 47 57 56
55 54 53 52 51 41 31 21 22 23 24 25
26 36 46 45 44 43 42 32 33 34 35



① top wall → minR ③ bottom wall → maxR

② right wall → maxC ④ left wall → minC

① Top wall, (minR, minC to maxC)

```
for(let c = minC; c<=maxC; c++) {
    psw(mat[minR][c]);
minR++;
```

(once the job is completed move to next top wall)

② right wall, (maxC, minR to maxR)

```
for(let r = minR; r <= maxR; r++) {
    psw(mat[r][maxC]);

maxC--;
```

③ bottom wall, (maxR, maxC to minC)

```
for(let c = maxC; c>=minC; c--) {
    psw(mat[maxR][C]);
}
maxR--;
```

④ leftwall (minC, maxR to minR)

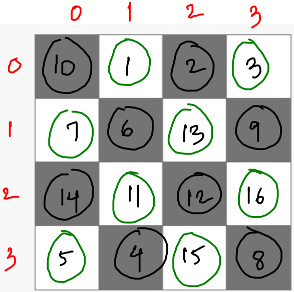
```
for(let r = maxR; r >= minR; r--) {
    psw(mat[r][minC]);
}
minC++;
```

★ TC : $O(N^2)$

SC : $O(1)$

Check how many times each ele is accessed/touched, here it is only once.

* Alternate Matrix Sum :



4 x 4 Chess Board

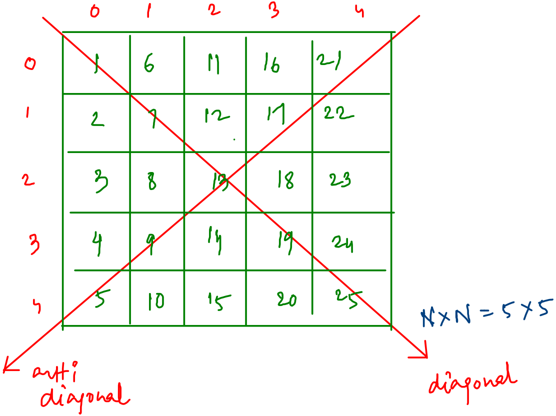
10 + 2 + 6 + 9 + 14 + 12
+ 4 + 8 = bsum

equal or not

1 + 3 + 7 + 13 + 11 + 16
+ 5 + 15 = wsum

black	white
(0, 0) = 0	(0, 1) = 1
(0, 2) = 2	(0, 3) = 3
(1, 1) = 2	(1, 0) = 1
(1, 3) = 4	(1, 2) = 3
(2, 0) = 2	(2, 1) = 3
(2, 2) = 4	(2, 3) = 5
(3, 1) = 4	(3, 0) = 3
(3, 3) = 6	(3, 2) = 5
even	odd

★ Diagonal Difference



diag = 1+7+13+19+25 =

adiag = 21+17+13+9+5 =

abs(diag − adiag) =

diag	anti diagonal
(0, 0)	(0, 4) ⇒ (0, 5−0−1)
(1, 1)	(1, 3) ⇒ (1, 5−1−1)
(2, 2)	(2, 2) ⇒ (2, 5−2−1)
(3, 3)	(3, 1) ⇒ (3, 5−3−1)
(4, 4)	(4, 0) ⇒ (4, 5−4−1)
r = c (r, r)	(r, n−r−1)

```
for(let r=0 ; r<n ; r++) {
    diag += mat[r][r];
    adiag += mat[r][n-r-1];
}
```