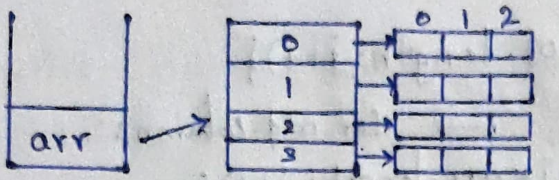
2D Arrays:

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

→ create 4 arrays of size 3.



2D Array:

Array of arrays.

So, arr.length = 4 → no. of rows

arr[0].length = 3 → no. of columns.

Abstract View:

arr →

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

Matrix Traversal:

→ Row

column ↓

1	2	3
4	5	6
7	8	9

Row Traversal:

O/P: 1, 2, 3, 4, 5, 6, 7, 8, 9

→ we want to go to each row and across each of its element.

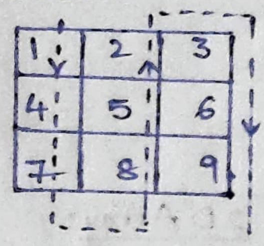
```
for (int i = 0; i < arr.length; i++){
  for (int j = 0; j < arr[0].length; j++){
     s.o.pln (arr[i][j]);
  }
}
```

→ **Columnar Traversal:**

O/P: 1, 4, 7, 2, 5, 8, 3, 6, 9.

```
for(int i = 0; i < arr[0].length; i++){
   for (int j = 03 j < arr.length; j++){
       s.o.pln (arr[j][i]);
   }
}
```

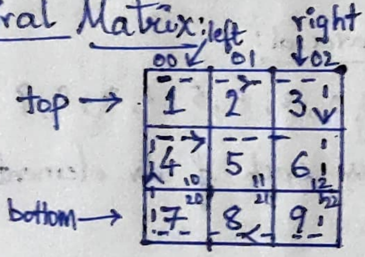
Wave Traversal:



o/p :- 1, 4, 7, 8, 5, 2, 3, 6, 9.

```
for (int j=0; j<arr[0].length; j++){
    if(j%2 ==0){  →  for odd columns
      for (int i=0; i<arr.length; i++){
          list.add( arr[i][j]);
      }
    }
    else{  →  for even columns.
      for (int i= arr.length; i>=0; i--){
          list.add (arr[i][j]);
      }
    }
}
```

Spiral Matrix:



```
int left =0; int right = arr[0].length-1;
int top = 0; int bottom = arr.length -1;
int direction = 0;
while (top<= bottom && left <= right){
    if(direction ==0){
      for(int i =left; i<=right; i++){
        S.o.pln (arr[top][i]);
      } top ++;
    }
if (direction==1){
for(int i=top; i<=bottom; i++){
    S.o.pln (arr[i][right]);
  }
  right--;
}
 if (direction == 2){
for (int i= right; i>=left; i--){
    S.o.pln (arr [bottom] [i]);
 }
 } bottom --;
 }
          if(direction ==3){
          for(int i= bottom; i>=top; i--){
             S.o.pln (arr [i][left);
          } left ++;
          }
          direction = (direction+1)%4;
          }
```

Matrix Addition:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} + \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = \begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \\ 14 & 16 & 18 \end{bmatrix}$$

```
int[][]ansArr = new int [arr.length][arr[0].length];
  for(int i=0; i<ansArr.length; i++){
  for (int j=0; j<ansArr[i].length; j++){
     ansArr [i][j] = ~~ansArr~~ arr1[i] [j] + arr2[i][j];
     }
  }
```

Matrix Multiplication:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} * \begin{bmatrix} 2 & 2 \\ 2 & 2 \\ 2 & 2 \end{bmatrix} = \begin{bmatrix} 1\times2+2\times2+3\times2 & 1\times2+2\times2+3\times2 \\ 4\times2+5\times2+6\times2 & 4\times2+5\times2+6\times2 \\ 7\times2+8\times2+9\times2 & 7\times2+8\times2+9\times2 \end{bmatrix}$$

$$= \begin{bmatrix} 12 & 12 \\ 50 & 30 \\ 48 & 48 \end{bmatrix}$$

```
int[][]ansArr= new int [arr1.length] [arr2[0].length];
   for(int i=0; i< ans Arr. length; i++){
   for (int j=0, i< ans Arr[0].length; j++){
     for (int k =0; k< arr1[0].length; k++){
          ans Arr [i] [j]= ansArr [i][j] + arr1[i][k]* arr2[k][j];
        }
     }
  }
```

→ **Search in 2d Sorted Array:**

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad \text{target} = 7$$

T.C = O(max (rows, columns)).

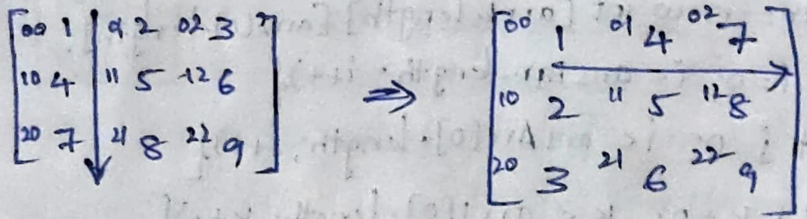
Algorithm:

→ Take a pointer on the top right corner and start comparing with the target's value.

→ If the value is less then increment the row value, else decrement the column value.

```
int i=0; int j= arr[0].length-1;
while(i<arr.length && j>=0){
    if (arr[i] [j] ==target){
        return true;
    }
    else if (arr[i] [j] <target){
        i++;
    }
    else
        j--;
}
return false;
```

→ **Transponse of a Matrix:**



```
int[][] ansArr = new int[arr[0].length] [arr.length];
  for (int i= 0; i< ansArr.length; i++){
    for (int j=0; j<ansArr[0].length; j++){
        ansArr[i] [j] = arr [j] [i]
    }
  }
```

Rotate Image: → rotate the image by 90°

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \Rightarrow \begin{bmatrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{bmatrix}$$

→ You are given nxn 2D. array representing an Arrays/Image rotate the image by 90° clockwise.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \xrightarrow{\text{Take Transpose}} \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

↓

Reverse of each row

$$\begin{bmatrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{bmatrix}$$

Algorithm:

Step1:- Transpose of matrix. Step2:- Reverse the matrix each row.

Reverse of array arr = [1, 2, 3, 4, 5]

o/p = [5, ~~4~~ 4, 3, 2, 1]

$$\begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \hline \boxed{1} & \boxed{2} & \boxed{3} & \boxed{4} & \boxed{5} \\ \uparrow & & & & \uparrow \\ \text{left} & & & & \text{right} \end{array}$$

```
while (left < right)
    {
        int temp = arr[left]
        arr [left] = arr [right]
        arr [right] = temp.
    }
    left ++;
    right--;
```

→ So, here also for the given problem we need to get transpose first

→ Then, we need to swap the elements of each row.

~~int[][] ansArr = new int[arr[0].length][arr.length];~~

~~for(int i=0; i<ansArr.length; i++){~~

~~for(int j=0; j<ansArr[0].length; j++){~~

Step ①:

```
for (int i=0; i<arr.length; i++){
  for (int j=0; j<arr[0].length;j++){
        int temp = arr[i][j];
         arr[i][j]= arr[j][i];
         arr[j][i] = temp;
     }
  }
```

Step ②:

```
    for (int  i=0; i<arr.length; i++){
         int left =0;
         int right = arr[0].length-1;
        while (left<right){
            int temp = arr[i][left]
            arr[i][left]= arr[i][right]
            arr[i][right] = temp;
            left ++;
            right -;
          }
     }
```