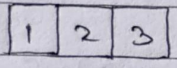
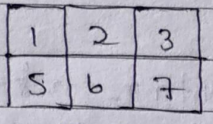
★ 2D-ARRAYS

Page No. 75
Date: / /

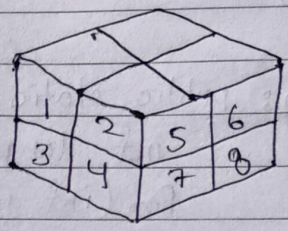
* 2-Dimensional Arrays.
* Data are stored in 2 directions.
* It is in form of Rows (→) & columns (↓).

* 

1-D

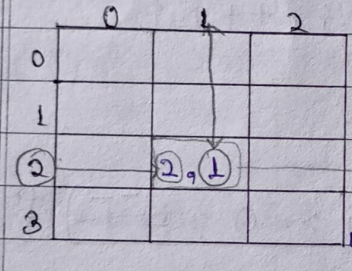


2-D



n-D

* Representation



→ Total cells: n x m
rows x columns
2, 1
2 x 1

Rows = 4
Columns = 3

→ Index

* Creation of 2D Arrays

⇒ datatype name [][] = new datatype [row][cols];

```
psvm {
    int matrix [][] = new int [3] [3];
    int n = matrix.length , m = matrix [0]. length;
                // 3 - row size          // 3 column size
    Scanner sc = new Scanner (System.in);
    for (int i=0; i<n; i++) {
        for (int j=0; j<m; j++) {
            matrix [i] [j] = sc.nextInt ();
        }
    }
```

Creation

Input

```
for (int i=0; i<n; i++) {
    for (int j=0; j<m; j++) {
        System.out.print (matrix[i][j] + "  ");
    }
    Syso();
}
```

Output

* Searching in 2D

```
public static boolean search (int matrix[][], int key) {
    for (int i=0; i< matrix.length; i++) {
        for (int j=0; j< matrix[0].length; j++) {
            if (matrix[i][j] == key) {
                System.out.print("Found at index : ("+i+","+j+")");
                return true;
            }
        }
    }
    Syso ("Key not found.");
    return false;
}
```

// for clarity of output.

* 2D Arrays in Memory

There are two way of storing 2D-Arrays :

1. Row Major

0	1	2	3	4	5	6	7	8

2. Column Major

0	3	6
1	4	7
2	5	8

In JAVA, the both ways are not used.

* If

	students →		
marks ↓	1	2	3
	4	5	6
	7	8	9

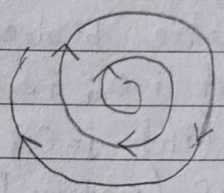
→

student1 →	1	2	3
student2 →	4	5	6
student3 →	7	8	9

In memory, actual data is stored in different 1D - Array.

* Spiral Matrix

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



Spiral.

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

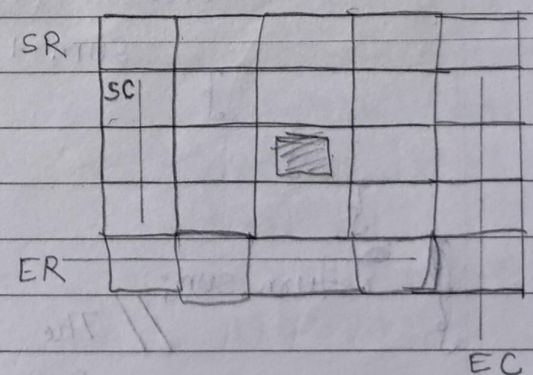
• Approach

	1st	2nd	3rd	
startRow	0	1	2	startRow ++
endRow	4 (n-1)	3	2	endRow --
startColumn	0	1	2	startColumn ++
endColumn	4 (n-1)	3	2	endColumn --

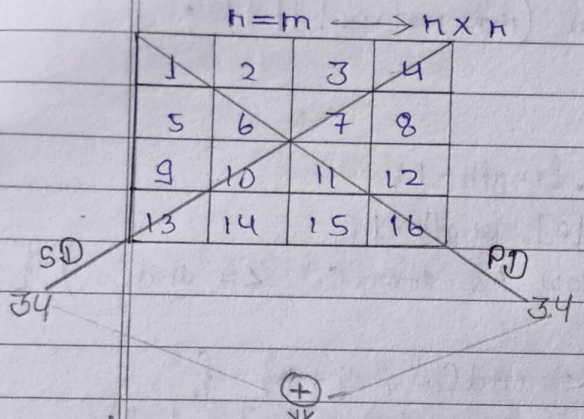
⇒ while loop ( ) {
print 1) top
2) right
3) bottom
4) left
}

Spiral Matrix - Code

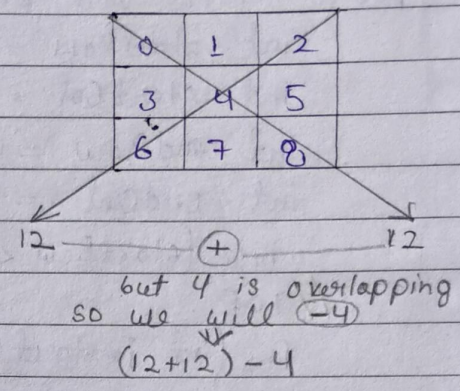
```
public static void printSpiral (int matrix[][]) {
    int startRow = 0;
    int startCol = 0;
    int endRow = matrix.length-1;
    int endCol = matrix[0].length-1;
    while (startRow <= endRow && startCol <= endCol) {
    //TOP
    for (int j=startCol; j<=endCol; j++) {
        System.out.print (matrix [startRow][j] + " ");
    }
    //RIGHT
    for (int i=startRow+1; i<=endRow; i++) {
        System.out.print (matrix [i][endCol] +" ");
    }
    //BOTTOM
    if(startRow == endRow) {
        break;
    }
    for (int j=endCol-1; j>=startCol; j--) {
        System.out.print (matrix [endRow][endCol]+" ");
    }
    //LEFT
    if(startCol == endCol)
    {
        break;
    }
    for (int i=endRow-1; i>=startRow+1; i--) {
        System.out.print (matrix [i][startCol]+" ");
    }
    //UPDATE
    startCol++;
    startRow++;
    endCol--;
    endRow--;
    }
    Syso();
}
```



* Diagonal Sum



Diagonal Sum : 68



Diagonal Sum: 20

PD : Primary Diagonal ($i = j$ always i.e, row = col)
SD : Secondary Diagonal ($i + j = n - 1$)

⇒ Diagonal sum Code $O(n^2)$

```
public static ~~void~~ int diagonal_sum (int matrix[][]) {
    int sum = 0;
    for (int i=0; i<matrix.length; i++) {
        for (int j=0; j<matrix[0].length; j++) {
            if (i==j) {
                sum += matrix[i][j];
            }
            else if (i+j == matrix.length-1) {
                sum += matrix[i][j];
            }
        }
    }
    return sum;
}
// The time complexity is not so efficient so we use another method to solve.
```

* for primary diagonal, → i == j then Sum.
* for secondary diagonal, → i + j = n-1
  → j = n-1-i then Sum.

⇒ Diagonal Sum code O(n)

```
public static ~~void~~ int diagonal_sum (int matrix[][]) {
    int sum = 0;
    for (int i=0; i< matrix.length; i++) {
        sum += matrix[i][i];
        if (i != matrix.length - 1 - i) {
            sum += matrix[i][matrix.length - i - 1];
        }
    }
    return sum;
}
```

* Search in Sorted Matrix

10	20	30	40
15	25	35	45
27	29	37	48
32	33	39	50

Staircase Search — Key = 33

10	20	30	40
15	25	35	45
27	29	37	48
32	33	39	50

C:1 (n-1, 0)
Key < cell value
TOP
Key > cell value
RIGHT

C:2 (0, m-1)
Key < cell value
LEFT
Key > cell value
BOTTOM

* Search in Sorted Matrix Code

C:2
```
public static ~~void~~boolean staircase_search (int matrix[][],
                                                int key) {
    int row = 0, col = matrix[0].length-1;
    while (row < matrix.length && col >= 0) {
        if (matrix[row][col] == key) {
            System.out.print ("Found key at ("+row+1+","
                              +col+1+")");
            return true;
        }
        else if (key < matrix[row][col]) {
            col--;
        }
        else {
            row++;
        }
    }
    System.out.println ("Key not Found !!!");
    return false;
}
```

// Time Complexity, O(m+n).

C:1
```
int row = matrix.length-1, col = 0;
while (row >=0 && col < matrix.length) {
    //same
    else if (key < matrix[row][col]) {
        row--;
    }
    else {
        col++;
    }
//same
```

QUESTIONS

Ques 1. Print the number 7's that are in the 2d Array.

Ex: int arr[][] = {{4, 7, 8}, {8, 8, 7}};

Output - 2.

```
public static void count_num (int matrix[][], int num) {
    int row = matrix.length, col = matrix[0].length;
    int count = 0;
    for (int i=0; i<row; i++) {
        for (int j=0; j<col; j++) {
            if (matrix[i][j] == num) {
                count++;
            }
        }
    }
    System.out.println("No. of times " +num+ " appears in
                        2D Array is : " + count);
}
```

Ques 2. Print out the sum of the numbers in the given specific row. of the array.

```
~~print~~ public static void sum_of_row (int matrix[][], int row_no){
    int sum = 0;
    for (int i=0; i<matrix[row_no].length, i++) {
        sum += matrix[row_no][i];
    }
    System.out.print ("sum of " +row_no+ " row
                       is " + sum);
}
```

Ques 3. Program of Transpose of a Matrix.

```
public static void print_matrix (int matrix[][]) {
    for (int i=0; i< matrix.length; i++) {
        for (int j=0; j < matrix.length; j++) {
            system.out.print (matrix[i][j] + " ");
        }
        syso ();
    }
}
public static void transpose (int matrix[][]) {
    int row = matrix.length;
    int cols = matrix[0].length;
    int transpose_matrix [][] = new int [row][cols];
    for (int i=0; i<row; i++) {
        for (int j=0; j< cols; j++) {
            transpose_matrix [j][i] = matrix[i][j];
        }
    }
    syso (" Transpose Matrix is: ");
    print_matrix (transpose_matrix);
}
```