2D Array

In [ ]:

```

```

What is a Matrix/2D array

In [ ]:

```

```

Language representation

In [ ]:

```

```

In [ ]:

```
Fill a matrix row wise and column wise
```

Diagonals

- main : i == j
- off: (i + j) == (n-1)

Triangles

- main: i <= j or i >=j
- off: (i+j) <= (n-1) or (i+j) >= (n-1)

In [ ]:

```
int mat[2][3];
//   0  1  2
// 0 10 20 30
// 1 40 50 60

mat[0][0] = 20
a = mat[1][2]

mat[0]
```

In [ ]:

```
1. int mat[2][3]; // static matrix  of size 2x3 => 6
2. vector<vector<int>> mat;
    vector<int> row = {};
    mat.push_back(row);
    vector<int> row = {};
    mat.push_back(row);

    [ []
      [] ] 2x0

    vector<vector<int>> mat;
    vector<int> row = {10,20,30};
    mat.push_back(row);
    vector<int> row = {10,20,30};
    mat.push_back(row);

    [ [10, 20, 30]
      [10, 20, 30] ] 2x3

    mat[0].push_back(40); // this is no longer a matrix after this op

Python
mat = []
mat.append([10,20,30])
mat.append([10,20,30])

    [ [10, 20, 30]
      [10, 20, 30] ] 2x3

C++
int **p;
p = new *int[2];
p[0] = new int[3];
p[1] = new int[3];

```

In [2]:

```

mat = []
mat.append([10,20,30])
mat.append([10,20,30])
print(mat)
mat[0][0] = 100
print(mat)
```

```
[[10, 20, 30], [10, 20, 30]]
[[100, 20, 30], [10, 20, 30]]
```

In [ ]:

```

```

In [ ]:

```
1. Create a matrix of size 3x5
 and populate the matrix with numbers 1....15
     a. row wise
     b. col wise


```

Row order

```
int main() {
    int mat[3][5];

    int value = 1;
    for(int i = 0; i < 3; i++) {
        for(int j = 0; j < 5; j++) {
            mat[i][j] = value++;
        }
    }

    for(int i = 0; i < 3; i++) {
        for(int j = 0; j < 5; j++) {
            cout << mat[i][j] << " ";
        }
        cout << endl;
    }


    cout << endl;

    vector<vector<int> > mat2;
    value = 1;
    for(int i = 0; i < 3; i++) {
        mat2.push_back(vector<int> ());
        for(int j = 0; j < 5; j++) {
             mat2[i].push_back(value++);
        }
    }


    for(int i = 0; i < 3; i++) {
        for(int j = 0; j < 5; j++) {
             cout << mat2[i][j] << " ";
        }
        cout << endl;
    }

}
```

```
matrix = []
counter = 1;
for row in range(0, 3):
    matrix.append([])
    for col in range(0, 5):
        matrix[row].append(counter)
        counter += 1

print(matrix)
```

```
// "static void main" must be defined in a public class.
public class Main {
    public static void main(String[] args) {
        int n=1;
        int[][] mat=new int[3][5];
        for(int i=0; i<mat.length; i++){
            for(int j=0; j<mat[0].length; j++){
                mat[i][j]=n;
                System.out.print(mat[i][j] + " ");
                n++;
            }
            System.out.println("");

        }
    }
}
```

```
public class matrix {
    public static void main(String[] args) {
        int[][] first = new int[3][5];

        int num = 1;
        for (int r = 0; r < 3; r++) {
            for (int c = 0; c < 5; c++) {
                first[r][c] = num++;
            }
        }

        for (int r = 0; r < 3; r++) {
            for (int c = 0; c < 5; c++) {
                System.out.println(first[r][c]);
            }
        }
    }
}
```

~~~~
**Column Order**
```C++
int main() {
    int mat[3][5];
~~~~

~~~

    int value = 1;
    for(int j = 0; j < 5; j++) {
        for(int i = 0; i < 3; i++) {
            mat[i][j] = value++;
        }
    }

    for(int i = 0; i < 3; i++) {
        for(int j = 0; j < 5; j++) {
            cout << mat[i][j] << " ";
        }
        cout << endl;
    }
}
```
Row Wise: Traverse columns first
Column Wise: Traverse row first

//   0 1 2 3 4
// 0 1 4
// 1 2 5
// 2 3 6

~~~

**Transpose: row <-> column**

```
10, 20, 30
40, 50, 60  2x3

10, 40
20, 50
30, 60 3x2
```

[https://leetcode.com/problems/transpose-matrix/ (https://leetcode.com/problems/transpose-matrix/)](https://leetcode.com/problems/transpose-matrix/)

~~~
**C++**
```C++
class Solution {
public:
    vector<vector<int>> transpose(vector<vector<int>>& matrix) {
        vector<vector<int>> ans;
        for(int i=0;i<matrix[0].size();i++){ // iterate column wise
            vector<int> v;
            for(int j=0;j<matrix.size();j++){ // row
                v.push_back(matrix[j][i]);
            }
            ans.push_back(v);
        }
        return ans;
    }
};
~~~

~~~~
```

**Python**
```Python
    b=matrix
    s=[ ]
    for i in range(len(b[0])):
        x=[]
        for j in range(len(b)):
            x.append(b[j][i])
        s.append(x)
    return s
  0 1 2
0 1 2 3
1 4 5 6
2x3
i 0..2
j 0..1
[[1  4]
 [2  5]
 [3  6]]

class Solution:
    def transpose(self, matrix: List[List[int]]) -> List[List[int]]:
        rows = len(matrix)
        cols = len(matrix[0])
        nm = [[0] * rows for _ in range(cols)] # pre-allocate

        for i in range(rows):
            for j in range(cols):
                nm[j][i] = matrix[i][j]
        return nm

    2x4  i=0..1   j=0..3
    4x2
```

**Java**
```Java
class Solution {
    public int[][] transpose(int[][] matrix) {
        int a = matrix[0].length;
        int b = matrix.length;

        int[][] res = new int[a][b];
        for (int i=0; i < a; i++){
            for(int j = 0; j < b; j++){
                res[i][j] = matrix[j][i];
            }
        }
        return res;
    }
}
```
~~~~

In [ ]:

Square matrix

In [ ]:

```
Rows==Columns

2x2, 3x3


1 2 3
4 5 6
7 8 9 3x3

1 4 7
2 5 8
3 6 9 3x3

nxn Matrix

//    0  1  2  i+j
// 0 00 01 02  0 1 2
// 1 10 11 12  1 2 3
// 2 20 21 22  2 3 4

// lower left (i>=j)
1
2 5
3 6 9

// Upper left (i+j) <= (n-1)
1 4 7
2 5
3

// Upper right (j>=i)
1 4 7
  5 8
    9


// Lower right (i+j) >= (n-1)
    7
  5 8
3 6 9
```

Diagonals

- main : i == j
- off: (i + j) == (n-1)

Triangles

- main: i <= j or i >=j
- off: (i+j) <= (n-1) or (i+j) >= (n-1)

Given a square matrix, transpose it in-place

```
class Solution {
public:
    void transpose(vector<vector<int>>& matrix) {
        for(int i=0;i<matrix.size();i++){

            for(int j=0;j<i;j++){
                int temp = matrix[j][i];
                matrix[j][i] = matrix[i][j];
                matrix[i][j] = temp;
            }
        }
        return matrix;
    }
};
```

```
   0 1 2
0  1 4 3
1  2 5 8
2  7 6 9
```

Question-1

https://leetcode.com/problems/rotate-image/ (https://leetcode.com/problems/rotate-image/)

In [ ]:

Question-2

https://leetcode.com/problems/spiral-matrix-ii/ (https://leetcode.com/problems/spiral-matrix-ii/)

In [ ]:

Question-3

https://leetcode.com/problems/search-a-2d-matrix/ (https://leetcode.com/problems/search-a-2d-matrix/)

```
O(m*n)
        for(auto i  =0 ; i < matrix.size(); i++) {
            for (auto j = 0; j < matrix[0].size(); j++) {
                if (matrix[i][j] == target) return true;
            }
        }
        return false;

        O(m*n)
        for(auto i  =0 ; i < matrix.size(); i++) {
            for (auto j = 0; j < matrix[0].size(); j++) {
                if (matrix[i][j] == target) return true;
                else if (matrix[i][j] > target) break;
            }
        }
        return false;

O(m log n)
        loop over all the rows // O(m)
        if target is in range of current row
           do binary search in current row // O(log n)

        O(log m + log n)
        binary search for the row // O(log m)
        while (srow <= erow)
          mid = (srow + erow)/2
          if (target >= matrix[mid][0] && target <= matrix[mid][matri
x[0].size()-1])
             do binary search in current row // O(log n)
             break
          else if (target < matrix[mid][0] ) // O(1)
             erow = mid -1
          else  O(1)
            srow = mid+1

O(log m + log n)= O( log m*n )
        binary search for the row // O(log m)

        Find the row where target might be present. O(log m)
        targetRow = -1
        while (srow <= erow)
          mid = (srow + erow)/2
          if (target >= matrix[mid][0] && target <= matrix[mid][matri
x[0].size()-1])
             targetRow = mid
             break
          else if (target < matrix[mid][0] ) // O(1)
```

```
             erow = mid -1
          else  O(1)
            srow = mid+1
        if (srow>erow) return false
        O(log n)
        while(scol <= ecol)
             mid = (scol+ecol)/2
             if (matrix[targetRow][mid] == target) return true
             else if (target < matrix[targetRow][mid]) ecol = mid -1
             else scol = mid + 1

O (log m*n )
        think of the 2d array as a 1 d array and convert the 1D index
to row and col.
```

In [ ]:

```

```

In [ ]:

```
# Binary search
```

```

```

DIY:

1. [basic] WAP to input m,n and input matrix and store data
2. [basic] WAP to input a matrix using above code and print sum of all numbers
3. [basic] WAP to input a matrix and print main diagonal elements
4. [basic] WAP to print all triangles of a square matrix
5. [basic] WAP to input a matrix using above code and print main diagonal elements
6. [basic] WAP to input two matrix m1, m2 and print their ***matrix*** multiplication.