DSA SERIES

- Learn Coding

Questions to be Covered today

1. **Spiral matrix**
2. **Spiral matrix II**
3. **Single Number**

LETS START TODAY'S LECTURE

LEETCODE - 54

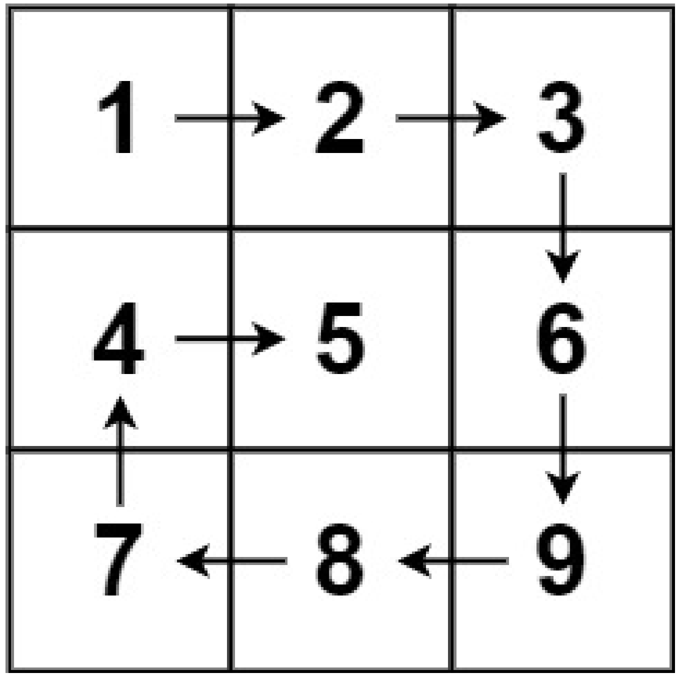
Spiral matrix

Given an m x n matrix, return *all elements of the* matrix *in spiral order*.

Example 1:

Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]

Output: [1 , 2 , 3 , 6 , 9 , 8 , 7 , 4 , 5]



Code :

```
class Solution {
public:
    vector<int> spiralOrder(vector<vector<int>>& matrix) {
        int m = matrix.size();
        int n = matrix[0].size();

        vector<int> result;

        int top = 0;
        int down = m - 1;

        int left = 0;
        int right = n - 1;

        int id = 0; // to know about the direction
```

Code :

```
while (top <= down && left <= right) {

     if (id == 0) { // left  to right
        for (int i = left; i <= right; i++) {
           result.push_back(matrix[top][i]);
        }
        top++;
     }

     if (id == 1) { // top to down
        for (int i = top; i <= down; i++) {
           result.push_back(matrix[i][right]);
        }
        right--;
     }

     if (id == 2) { // right to left
        for (int i = right; i >= left; i--) {
           result.push_back(matrix[down][i]);
        }
        down--;
     }
```

```
 if (id == 3) { // down to top
               for (int i = down; i >= top; i--) {
                   result.push_back(matrix[i][left]);
               }
               left++;
           }

           id = (id + 1) % 4;
       }

       return result;
   }
};
```

LEETCODE - 59

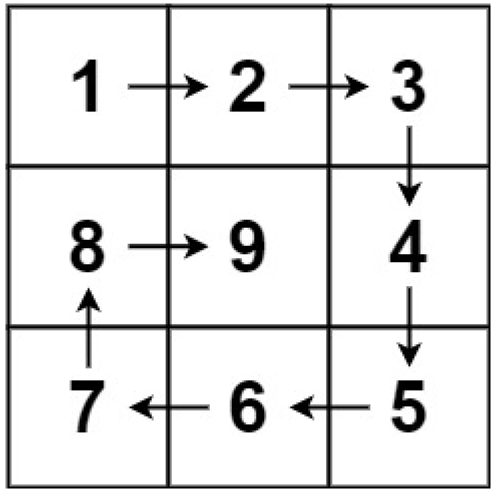
Spiral Matrix II

Given a positive integer n, generate an n x n matrix filled with elements from 1 to n^2 in spiral order.

Example 1:

Input: n = 3

Output: [[1,2,3],[8,9,4],[7,6,5]]



Code :

```
class Solution {
public:
    vector<vector<int>> generateMatrix(int n) {
        vector<vector<int>> result(n,
vector<int>(n));

        int top = 0;
        int down = n - 1;

        int left = 0;
        int right = n - 1;

        int id = 0; // direction
        int num = 1;
```

```
while (top <= down && left <= right) {
           if (id == 0) { // left to right
               for (int i = left; i <= right; i++) {
                   result[top][i] = num++;
               }
               top++;
           }

           if (id == 1) { // top to down
               for (int i = top; i <= down; i++) {
                   result[i][right] = num++;
               }
               right--;
           }

           if (id == 2) { // right to left
               for (int i = right; i >= left; i--) {
                   result[down][i] = num++;
               }
               down--;
           }
```

```
 if (id == 3) { // down to top
               for (int i = down; i >= top; i--) {
                   result[i][left] = num++;
               }
                      left++;
           }

           id = (id + 1) % 4;
       }
       return result;
   }
};
```

LEETCODE - 136

Single Number

Given a **non-empty** array of integers nums, every element appears *twice* except for one. Find that single one.
You must implement a solution with a linear runtime complexity and use only constant extra space.

Example 1:
Input: nums = [2,2,1]
Output: 1

Example 2:
Input: nums = [4,1,2,1,2]
Output: 4

Example 3:
Input: nums = [1]
Output: 1

Code :

```
class Solution {
public:
    int singleNumber(vector<int>& nums) {
        int n = nums.size();

        int result = 0;

        for (int i = 0; i < n; i++) {
            result ^= nums[i];
        }

        return result;
    }
};
```

Learn coding

THANK YOU