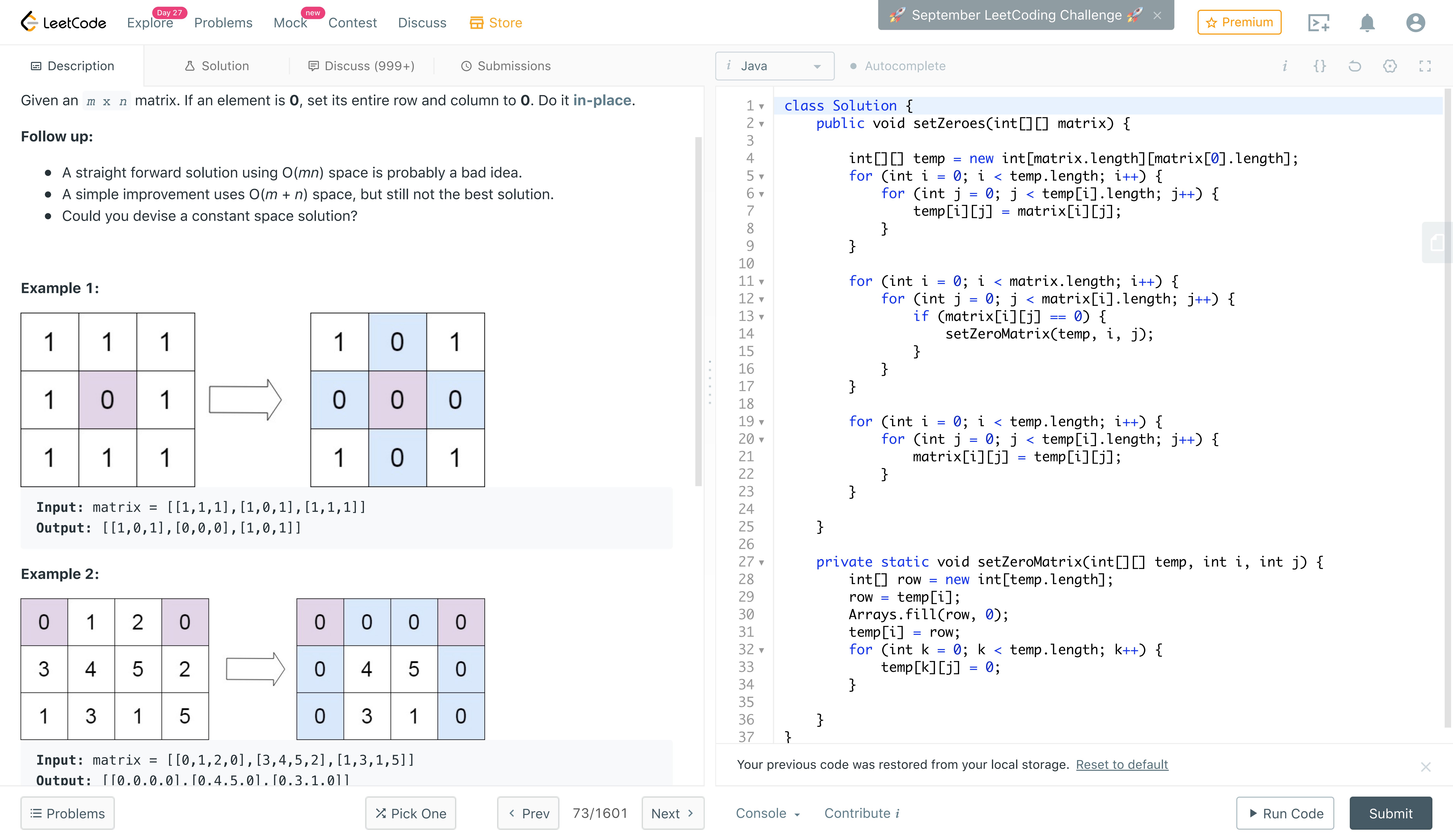

Given an `m x n` matrix. If an element is **0**, set its entire row and column to **0**. Do it **in-place**.

Follow up:

- A straight forward solution using O(*mn*) space is probably a bad idea.
- A simple improvement uses O(*m* + *n*) space, but still not the best solution.
- Could you devise a constant space solution?

Example 1:

1	1	1
1	0	1
1	1	1

1	0	1
0	0	0
1	0	1

```
Input: matrix = [[1,1,1],[1,0,1],[1,1,1]]
Output: [[1,0,1],[0,0,0],[1,0,1]]
```

Example 2:

0	1	2	0
3	4	5	2
1	3	1	5

0	0	0	0
0	4	5	0
0	3	1	0

```
Input: matrix = [[0,1,2,0],[3,4,5,2],[1,3,1,5]]
Output: [[0,0,0,0],[0,4,5,0],[0,3,1,0]]
```

```java
class Solution {
    public void setZeroes(int[][] matrix) {

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

        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[i].length; j++) {
                if (matrix[i][j] == 0) {
                    setZeroMatrix(temp, i, j);
                }
            }
        }

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

    }

    private static void setZeroMatrix(int[][] temp, int i, int j) {
        int[] row = new int[temp.length];
        row = temp[i];
        Arrays.fill(row, 0);
        temp[i] = row;
        for (int k = 0; k < temp.length; k++) {
            temp[k][j] = 0;
        }

    }
}
```

Your previous code was restored from your local storage. Reset to default