In the popular **Minesweeper** game you have a board with some mines and those cells that don't contain a mine have a number in it that indicates the total number of mines in the neighboring cells. Starting off with some arrangement of mines we want to create a **Minesweeper** game setup.

**Example**

For

matrix = [[true, false, false],

[false, true, false],

[false, false, false]]

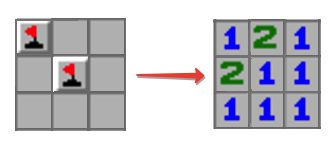
the output should be

minesweeper(matrix) = [[1, 2, 1],

[2, 1, 1],

[1, 1, 1]]

Check out the image below for better understanding:



**Input/Output**

* **[time limit] 3000ms (cs)**
* **[input] array.array.boolean matrix**

A non-empty rectangular matrix consisting of boolean values - true if the corresponding cell contains a mine, false otherwise.

*Constraints:*  
2 ≤ matrix.length ≤ 5,  
2 ≤ matrix[0].length ≤ 5.

* **[output] array.array.integer**

Rectangular matrix of the same size as matrixeach cell of which contains an integer equal to the number of mines in the neighboring cells. Two cells are called neighboring if they share at least one corner.

<https://codefights.com/arcade/code-arcade/waterfall-of-integration/ZMR5n7vJbexnLrgaM>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static int[][] minesweeper(bool[][] matrix)

{

int[][] res = new int[matrix.Length][];

for (int i = 0; i < matrix.Length; i++)

{

res[i] = new int[matrix[i].Length];

for (int j = 0; j < matrix[i].Length; j++)

{

res[i][j] = 0;

}

}

for (int i = 0; i < matrix.Length; i++)

{

for (int j = 0; j < matrix[i].Length; j++)

{

int minasAdyacentes = 0;

if (i - 1 >= 0 && j - 1 >= 0)

{

if (matrix[i - 1][j - 1])

{

minasAdyacentes++;

}

}

if (i - 1 >= 0)

{

if (matrix[i - 1][j])

{

minasAdyacentes++;

}

}

if (i - 1 >= 0 && j+1 < matrix[i].Length)

{

if (matrix[i - 1][j + 1])

{

minasAdyacentes++;

}

}

if (j - 1 >= 0)

{

if (matrix[i][j - 1])

{

minasAdyacentes++;

}

}

//if (matrix[i][j])

//{

// minasAdyacentes++;

//}

if (j + 1 < matrix[i].Length)

{

if (matrix[i][j + 1])

{

minasAdyacentes++;

}

}

if (i + 1 < matrix.Length &&j-1>=0)

{

if (matrix[i + 1][j - 1])

{

minasAdyacentes++;

}

}

if (i + 1 < matrix.Length)

{

if (matrix[i + 1][j])

{

minasAdyacentes++;

}

}

if (i + 1 < matrix.Length && j + 1 < matrix[i].Length)

{

if (matrix[i + 1][j + 1])

{

minasAdyacentes++;

}

}

res[i][j] = minasAdyacentes;

}

}

return res;

}

static void Main(string[] args)

{

bool[][] matrix = {new bool[]{true, false, false},

new bool[]{false, true, false},

new bool[]{false, false, false}};

int[][] res = minesweeper(matrix);

for (int i = 0; i < res.Length; i++)

{

for (int j = 0; j < res[i].Length; j++)

{

Console.Write(res[i][j] + " ");

}

Console.WriteLine();

}

Console.ReadLine();

}

}

}