**Cavity Map**

**by**[**Gera1d**](https://www.hackerrank.com/Gera1d)

* [**Problem**](https://www.hackerrank.com/challenges/cavity-map)
* [**Submissions**](https://www.hackerrank.com/challenges/cavity-map/submissions)
* [**Leaderboard**](https://www.hackerrank.com/challenges/cavity-map/leaderboard)
* [**Discussions**](https://www.hackerrank.com/challenges/cavity-map/forum)
* [**Editorial**](https://www.hackerrank.com/challenges/cavity-map/editorial)
* [**Topics**](https://www.hackerrank.com/challenges/cavity-map/topics)

You are given a square map of size . Each cell of the map has a value denoting its depth. We will call a cell of the map a *cavity* if and only if this cell is not on the border of the map and each cell adjacent to it has *strictly smaller depth*. Two cells are adjacent if they have a common side (edge).

You need to find all the cavities on the map and depict them with the uppercase character **X**.

**Input Format**

The first line contains an integer, , denoting the size of the map. Each of the following  lines contains  positive digits without spaces. Each digit (1-9) denotes the depth of the appropriate area.

**Constraints** 

**Output Format**

Output  lines, denoting the resulting map. Each cavity should be replaced with character X.

**Sample Input**

4

1112

1912

1892

1234

**Sample Output**

1112

1X12

18X2

1234

**Explanation**

The two cells with the depth of 9 fulfill all the conditions of the Cavity definition and have been replaced by X.

**Related Topics**

* [String Basics](https://www.hackerrank.com/challenges/cavity-map/topics/string-basics)

**Submissions:**

[27680](https://www.hackerrank.com/challenges/cavity-map/leaderboard)

**Max Score:**

30

**Difficulty:**

Easy

More

[C#](javascript:void(0))





1

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static void Main(String[] args)

{

int n = Convert.ToInt32(Console.ReadLine());

string[] grid = new string[n];

for (int grid\_i = 0; grid\_i < n; grid\_i++)

{

grid[grid\_i] = Console.ReadLine();

}

string[] answer = new string[n];

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

{

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

{

int contMenores = 0;

if (i - 1 >= 0)

{

if (int.Parse(grid[i][j].ToString()) > int.Parse(grid[i - 1][j].ToString()))

{

contMenores++;

}

}

if (i + 1 < grid.Length)

{

if (int.Parse(grid[i][j].ToString()) > int.Parse(grid[i + 1][j].ToString()))

{

contMenores++;

}

}

if (j - 1 >= 0)

{

if (int.Parse(grid[i][j].ToString()) > int.Parse(grid[i][j-1].ToString()))

{

contMenores++;

}

}

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

{

if (int.Parse(grid[i][j].ToString()) > int.Parse(grid[i][j + 1].ToString()))

{

contMenores++;

}

}

if (contMenores == 4)

{

answer[i] += 'X';

}

else

{

answer[i] += grid[i][j];

}

}

}

foreach (string s in answer)

{

Console.WriteLine(s);

}

Console.ReadLine();

}

}

}

Line: 62 Col: 1

Run Code Submit Code

Upload Code as File

Test against custom input

**Congrats, you solved this challenge!**