**package com.leetcode.array;**

**import java.util.Arrays;**

**/\***

**\* 1929. Concatenation of Array**

**Given an integer array nums of length n, you want to create an array ans of length 2n where ans[i] == nums[i] and ans[i + n] == nums[i] for 0 <= i < n (0-indexed).**

**Specifically, ans is the concatenation of two nums arrays.**

**Return the array ans.**

**Example 1:**

**Input: nums = [1,2,1]**

**Output: [1,2,1,1,2,1]**

**Explanation: The array ans is formed as follows:**

**- ans = [nums[0],nums[1],nums[2],nums[0],nums[1],nums[2]]**

**- ans = [1,2,1,1,2,1]**

**Example 2:**

**Input: nums = [1,3,2,1]**

**Output: [1,3,2,1,1,3,2,1]**

**Explanation: The array ans is formed as follows:**

**- ans = [nums[0],nums[1],nums[2],nums[3],nums[0],nums[1],nums[2],nums[3]]**

**- ans = [1,3,2,1,1,3,2,1]**

**Constraints:**

**n == nums.length**

**1 <= n <= 1000**

**1 <= nums[i] <= 1000**

**\*/**

**public class Concatenation\_01 {**

**public int[] getConcatenation(int[] nums) {**

**//nums-->length=3**

**//ans=3\*2**

**//ans[i] == nums[i] and ans[i + n] == nums[i] for 0 <= i < n (0-indexed).**

**int n=nums.length;**

**int []ans=new int[nums.length\*2];**

**for(int i=0;i<nums.length;i++)**

**{**

**ans[i]=nums[i];**

**ans[i+n]=nums[i];**

**}**

**return ans;**

**}**

**public static void main(String[] args) {**

**/\***

**int nums[]; //array decleration**

**nums=new int[3];//array size initilization**

**\*/**

**// int nums[]=new int[3];//at the time of array declearation and size initilization**

**// int nums[]=new int[] {1,2,1};//decleration and initiliaztion**

**int [] nums= {1,2,1};**

**Concatenation\_01 c=new Concatenation\_01();**

**System.*out*.println(Arrays.*toString*(c.getConcatenation(new int []{1,2,1})));**

**System.*out*.println(Arrays.*toString*(c.getConcatenation(new int []{1,3,2,1})));**

**}**

**}**

06

[**2373. Largest Local Values in a Matrix**](https://leetcode.com/problems/largest-local-values-in-a-matrix/)

You are given an n x n integer matrix grid.

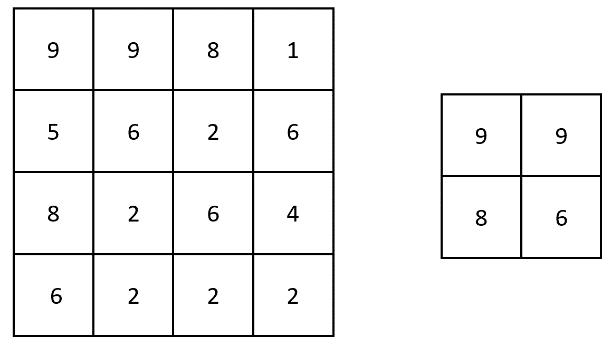
Generate an integer matrix maxLocal of size (n - 2) x (n - 2) such that:

* maxLocal[i][j] is equal to the **largest** value of the 3 x 3 matrix in grid centered around row i + 1 and column j + 1.

In other words, we want to find the largest value in every contiguous 3 x 3 matrix in grid.

Return *the generated matrix*.

**Example 1:**



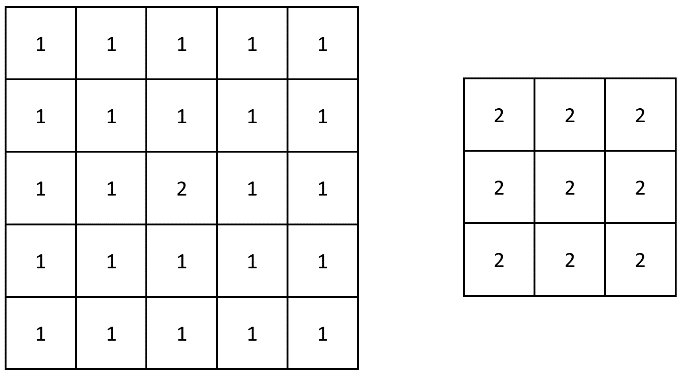
**Input:** grid = [[9,9,8,1],[5,6,2,6],[8,2,6,4],[6,2,2,2]]

**Output:** [[9,9],[8,6]]

**Explanation:** The diagram above shows the original matrix and the generated matrix.

Notice that each value in the generated matrix corresponds to the largest value of a contiguous 3 x 3 matrix in grid.

**Example 2:**



**Input:** grid = [[1,1,1,1,1],[1,1,1,1,1],[1,1,2,1,1],[1,1,1,1,1],[1,1,1,1,1]]

**Output:** [[2,2,2],[2,2,2],[2,2,2]]

**Explanation:** Notice that the 2 is contained within every contiguous 3 x 3 matrix in grid.

**Constraints:**

* n == grid.length == grid[i].length
* 3 <= n <= 100
* 1 <= grid[i][j] <= 100