**27. Remove Element**

Easy

19683381Add to ListShare

Given an array nums and a value val, remove all instances of that value [**in-place**](https://en.wikipedia.org/wiki/In-place_algorithm) and return the new length.

Do not allocate extra space for another array, you must do this by **modifying the input array**[**in-place**](https://en.wikipedia.org/wiki/In-place_algorithm) with O(1) extra memory.

The order of elements can be changed. It doesn't matter what you leave beyond the new length.

**Clarification:**

Confused why the returned value is an integer but your answer is an array?

Note that the input array is passed in by **reference**, which means a modification to the input array will be known to the caller as well.

Internally you can think of this:

// **nums** is passed in by reference. (i.e., without making a copy)

int len = removeElement(nums, val);

// any modification to **nums** in your function would be known by the caller.

// using the length returned by your function, it prints the first **len** elements.

for (int i = 0; i < len; i++) {

    print(nums[i]);

}

**Example 1:**

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

**Output:** 2, nums = [2,2]

**Explanation:** Your function should return length = **2**, with the first two elements of nums being **2**.

It doesn't matter what you leave beyond the returned length. For example if you return 2 with nums = [2,2,3,3] or nums = [2,2,0,0], your answer will be accepted.

**Example 2:**

**Input:** nums = [0,1,2,2,3,0,4,2], val = 2

**Output:** 5, nums = [0,1,4,0,3]

**Explanation:** Your function should return length = **5**, with the first five elements of *nums* containing **0**, **1**, **3**, **0**, and **4**. Note that the order of those five elements can be arbitrary. It doesn't matter what values are set beyond the returned length.

**Constraints:**

* 0 <= nums.length <= 100
* 0 <= nums[i] <= 50
* 0 <= val <= 100