**Remove Element**

Problem Statement: 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" \t "_blank)** with O(1) extra memory.

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

**Example 1:**

Given *nums* = **[3,2,2,3]**, *val* = **3**,

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.

**Example 2:**

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

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.

**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 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]);

}