Remove Element

# Question

Given an array nums and a value val, remove all instances of that value in-place and return the new length.

Do not allocate extra space for another array, you must do this by modifying the input array in-place 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.

**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.

# Pseudo Code

Declare and Initialize length variable to size - 1 and tempVariable to 0

Run the Outer For Loop from i=0 to length

If Element matches

Swap that Element with the one at position length

If the Swapped Element is still the same

Decrement the Value of i

Decrement the Value of length

Return (length + 1)

# Source Code

## v 1.0 (Language C)

1. int removeElement(int\* nums, int numsSize, int val){
3. int length = (numsSize - 1), temp = 0;
4. //making sure that the variable length points at the last index of the array.
6. for(int i=0 ; i<=length ; i++) {
8. if(nums[i] == val) {
9. temp = nums[i];
10. nums[i] = nums[length];
11. nums[length] = temp;
12. if(nums[i] == val)
13. i--;
14. length--;
15. }
16. }
17. return (length + 1);
18. }