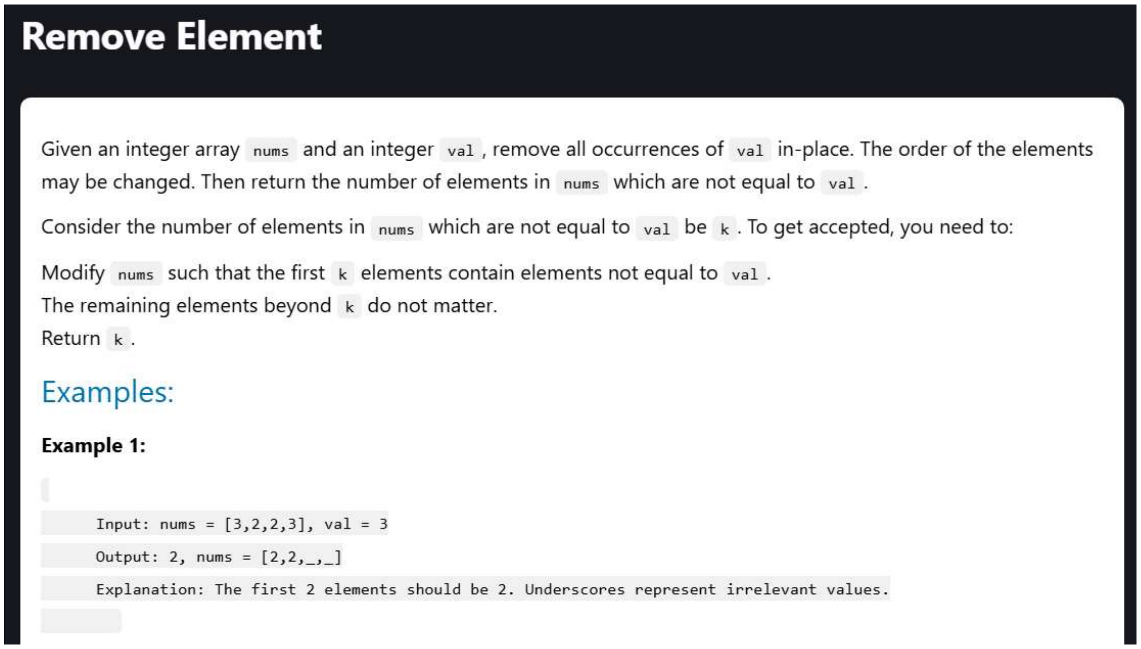
Remove Elements

Similar concept. Take a hole variable. And try to fill the value over there which is not equal to val. Once filled take it forward by increasing it.



Remove Element

Given an integer array `nums` and an integer `val`, remove all occurrences of `val` in-place. The order of the elements may be changed. Then return the number of elements in `nums` which are not equal to `val`.

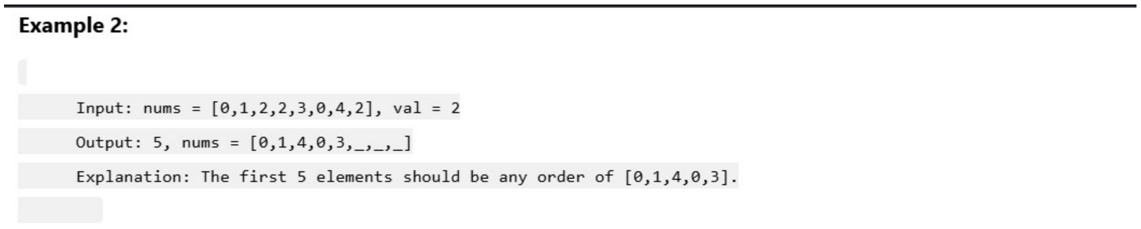
Consider the number of elements in `nums` which are not equal to `val` be `k`. To get accepted, you need to:

Modify `nums` such that the first `k` elements contain elements not equal to `val`.
The remaining elements beyond `k` do not matter.
Return `k`.

Examples:

Example 1:

```
Input: nums = [3,2,2,3], val = 3
Output: 2, nums = [2,2,_,_]
Explanation: The first 2 elements should be 2. Underscores represent irrelevant values.
```



Example 2:

```
Input: nums = [0,1,2,2,3,0,4,2], val = 2
Output: 5, nums = [0,1,4,0,3,_,_,_]
Explanation: The first 5 elements should be any order of [0,1,4,0,3].
```

Approach: Two Pointer Technique

Use pointer `x` to track where the next non-`val` element should go.
Traverse the array with index `i`.
If `nums[i] != val`, assign `nums[x] = nums[i]` and increment `x`.

Dry Run

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

i = 0 → nums[0] = 0 ≠ 2 → nums[0] = 0, x = 1
i = 1 → nums[1] = 1 ≠ 2 → nums[1] = 1, x = 2
i = 2 → nums[2] = 2 = 2 → skip
i = 3 → nums[3] = 2 = 2 → skip
i = 4 → nums[4] = 3 ≠ 2 → nums[2] = 3, x = 3
i = 5 → nums[5] = 0 ≠ 2 → nums[3] = 0, x = 4
i = 6 → nums[6] = 4 ≠ 2 → nums[4] = 4, x = 5
i = 7 → nums[7] = 2 = 2 → skip

Result: k = 5, nums = [0,1,3,0,4,...]
```

Complexity

Time: O(N)
Space: O(1)

JavaScript | C++ | C | Java | Python

```javascript
var removeElement = function(nums, val) {
    let x = 0;
    for (let i = 0; i < nums.length; i++) {
        if (nums[i] != val) {
            nums[x] = nums[i];
            x++;
        }
    }
    return x;
};
```