Remove Duplicates from Sorted Array

Given a sorted array nums, remove the duplicates in-place such that each element appear only once 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.

Example 1:

```
Given nums = [1,1,2],
Your function should return length = 2, with the first two elements of nums being 1 and 2 respectively.
It doesn't matter what you leave beyond the returned length.
```

Example 2:

```
Given nums = [0,0,1,1,1,2,2,3,3,4],
Your function should return length = 5, with the first five elements of nums being modified to 0, 1, 2, 3, and
4 respectively.
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 = removeDuplicates(nums);
// 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]);
}
```

Solution in C++

```
class Solution {
public:
    int removeDuplicates(vector<int>& nums) {
        map<int,int> mp;
        for(int i=0; i<nums.size(); i++) {
            if (!mp[nums[i]]) {
                mp[nums[i]]=i;
                nums[ mp.size()-1] = nums[i];
            }
        }

        return mp.size();
    }
};
```

Solution in Java

```
public int removeDuplicates(int[] nums) {
    if (nums.length == 0) return 0;
    int i = 0;
    for (int j = 1; j < nums.length; j++) {
        if (nums[j] != nums[i]) {
            i++;
            nums[i] = nums[j];
        }
    }
    return i + 1;
}
```