1.Easy\04.Remove_duplicates_from_sorted_array.cpp

```
/*
QUESTION:-

Given an integer array nums sorted in non-decreasing order, remove the duplicates in-place such that each unique element appears only once. The relative order of the elements should be kept the same. Then return the number of unique elements in nums.
Consider the number of unique elements of nums to be k, to get accepted, you need to do the following things:
Change the array nums such that the first k elements of nums contain the unique elements in the order they were present in nums initially. The remaining elements of nums are not important as well as the size of nums.
Return k.

Example 1:

Input: nums = [1,1,2]
Output: 2, nums = [1,2,_]
Explanation: Your function should return k = 2, with the first two elements of nums being 1 and 2 respectively.
It does not matter what you leave beyond the returned k (hence they are underscores).
Example 2:

Input: nums = [0,0,1,1,1,2,2,3,3,4]
Output: 5, nums = [0,1,2,3,4,_,_,_,_,_]
Explanation: Your function should return k = 5, with the first five elements of nums being 0, 1, 2, 3, and 4 respectively.
It does not matter what you leave beyond the returned k (hence they are underscores).
*/

/*
APPROACH:-
-> The idea, is to use keep a pointer k which signifies that upto here the array is sorted
-> Now travese the entire array and if arr[k]!=arr[j] that is arr[j] is a unique value hence it should be included
   so increment the k and swap arr[k] with arr[j]
-> Return k+1, +1 is because of 0 based indexing
*/

// CODE:-
int removeDuplicates(vector<int> &nums)
{
    int k = 0; // upto k array contains unique elements
    for (int j = 1; j < nums.size(); j++)
    {
        if (nums[k] != nums[j])
        {
            k++;
            swap(nums[k], nums[j]);
        }
    }
    return k + 1;
}
```