```csharp
class Program
{
    /*
     * 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.
     */
    static void Main(string[] args)
    {
        int[] nums = { 1, 2, 2, 3, 3, 3};

        Console.WriteLine("How many unduplicated elements are in the given array?");
        Console.WriteLine($"{RemoveDuplicates(nums)}");
    }


    /*
     * Approach - Two Pointers
     *
     * Since the array is already sorted, can keep two pointers i and j,
     * where i is the slow-runner while j is the fast-runner.
     * As long as nums[i] == nums[j], we increment j to skip the duplicate.
     *
     * 1. Compare the current element and the next element.
     * 2. If they are not equal, store the current element and exclude it from the array.
     * 3. Start thinking the next element as the first element,
     * and then compare the element to the next element again.
     * 4. Repeat the (2).
     */
    public static int RemoveDuplicates(int[] nums)
    {
        int i = 0;
        for(int j=1; j < nums.Length; j++)
        {
            if (nums[j] != nums[i])
            {
                i++;
                nums[i] = nums[j];
            }
        }
        return i + 1;
    }
}
```