LeetCode Problems:

1. #1 Two Sum
    a. This is perhaps the most popular LeetCode problem and one that I have done several times before. My idea behind this solution is to utilize an unordered list, as they can not contain duplicates (this is key). I then take the difference between the target and the current array position and assign that to the "dif" variable. We then look for dif in the array. If it is there, we return dif and the current position in the array; that is our answer. This solution is O(n).
        i.

```cpp
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target)
    {
        unordered_map <int,int> umap;

        for(int i=0;i<nums.size();i++)
        {
            int dif = target-nums[i];

            if(umap.find(dif)!=umap.end())
            {
                return{umap[dif], i};
            }

            umap[nums[i]] = i;

        }
        return {};
    }
};
```

2. #27 Remove Element
    a. For this problem, the approach I've taken is to create a variable "k" initialized to 0 that will be very important for removing "val" elements in place. We loop through the array, if the current element does NOT equal val, we will replace the next element with the current one; rinse and repeat until all val elements are removed. This is an O(n) solution.

i.

C++ Auto

```cpp
class Solution {
public:
    int removeElement(vector<int>& nums, int val) {
        int k = 0;
        for(int i=0;i<nums.size();i++)
        {
            if(nums[i]!= val)
            {
                nums[k++] = nums[i];
            }

        }
        return k;

    }
};
```

3. #26 Remove Duplicate From Sorted Array
    a. This problem is *very* similar to the previous one with a few key differences. Rather than having our "j" variable initialized to 0, we initialize it to 1, we also start iterating at the 1 position rather than 0. If the current element does not equal the one before it, the position at the jth element in the array is set to the ith position in the array. J is then incremented by one. Again, rinse and repeat.

i.

C++ Auto

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

            }
        }
        return j;
    }
};
```