```
// Remove Duplicates from sorted Array

class Solution {
public:
    int removeDuplicates(vector<int>& nums) {
        if (nums.empty()) return 0; // Edge case for empty array

    int j = 0; // Pointer for placing unique elements

    for (int i = 1; i < nums.size(); i++) {
        if (nums[i] != nums[j]) { // Found a new unique element
            j++; // Move to next position
            nums[j] = nums[i]; // Store the unique element
        }
    }

    return j + 1; // Number of unique elements
}
};

// Implementing Insertion Sort

class Solution {
public:
    int removeDuplicates(vector<int>& nums) {
        if (nums.empty()) return 0; // Edge case for empty array

    int j = 0; // Pointer for placing unique elements

    for (int i = 1; i < nums.size(); i++) {
        if (nums[i] != nums[j]) { // Found a new unique element
            j++; // Move to next position
            nums[j] = nums[i]; // Store the unique element
        }
    }

    return j + 1; // Number of unique elements
}
};

// Two Sum

class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {

    unordered_map<int, int> numMap; // Stores num -> index
```

```
    for (int i = 0; i < nums.size(); i++) {
        int complement = target - nums[i]; // The number we need to find

        // Check if the complement exists in the map
        if (numMap.find(complement) != numMap.end()) {
            return {numMap[complement], i}; // Return indices of the two numbers
        }

        // Store the current number and its index in the map
        numMap[nums[i]] = i;
    }

    return {}; // This line is never reached because a solution always exists
    }
};
```