***Question no. 1***

***Leet code Question no. 9***

***Question name:*** *Palindrome Number*

***Level:*** *Easy*

***Problem Description****:*

*The question asks us to check if a given number is a palindrome.*

*A palindrome is something that reads the same forward and backward.  
For example, numbers 121, 1331, and 12321 are palindromes because when reversed, they remain the same.*

*However, numbers like 123, 10, and -121 are not palindromes because reversing them gives a different number.*

***Code :***

*class Solution {*

*public:*

*bool isPalindrome(int x) {*

*if (x < 0) return false;*

*long long original = x, reversed = 0;*

*while (x > 0) {*

*reversed = reversed \* 10 + (x % 10);*

*x /= 10;*

*}*

*return original == reversed;*

*}*

*int main() {*

*int x = 998765432;*

*cout << (isPalindrome(x) ? "Palindrome" : "Not a palindrome") << endl;*

*return 0;*

*}*

*};*

***Question no. 2***

***Leet code Question no. 136***

***Question name:*** *Single Number*

***Level:*** *Easy*

***Problem Description****:*

*The question asks us to find a number that appears only once in a list where every other number appears twice.*

*For example, in [4, 1, 2, 1, 2], the number 4 appears only once, while 1 and 2 appear twice. So, the answer is 4.*

*However, in [2, 2, 1], the number 1 appears only once, so the answer is 1.*

*Our goal is to find this unique number* ***without using extra space*** *(like storing numbers in a separate list or map).*

***Code:***

*class Solution {*

*public:*

*int singleNumber(vector<int>& nums) {*

*int unique\_num = 0;*

*for (int i = 0; i < nums.size(); i++) {*

*unique\_num ^= nums[i];*

*}*

*return unique\_num;*

*}*

*};*

***Question no. 3***

***Leet code Question no. 1748***

***Question name:*** *Sum of Unique Elements*

***Level:*** *Easy*

***Problem Description****:*

*The question asks us to find the sum of unique elements in a list. A unique element is a number that appears only once in the list.*

*For example, in [1, 2, 3, 2], the numbers 1 and 3 appear only once, while 2 appears twice. So, the answer is 1 + 3* ***=*** *4.*

*However, in [4, 4, 4, 4], there are no unique numbers, so the answer is 0.*

*Our task is to find all numbers that appear only once and add them together to get the final sum.*

***Code:***

*class Solution {*

*public:*

*int sumOfUnique(vector<int>& nums) {*

*int sum = 0;*

*for (int i = 0; i < nums.size(); i++) {*

*bool isUnique = true;*

*for (int j = 0; j < nums.size(); j++) {*

*if (i != j && nums[i] == nums[j]) {*

*isUnique = false;*

*break;*

*}*

*}*

*if (isUnique) {*

*sum += nums[i];*

*}*

*}*

*return sum;*

*}*

*};*

***Question no. 4***

***Leet code Question no. 1***

***Question name:*** *Two sum*

***Level:*** *Easy*

***Problem Description:***

*The question asks us to find two numbers in a list that add up to a given target value. We need to return the indices of these two numbers.*

*For example, in [2, 7, 11, 15] with target = 9, the numbers 2 and 7 add up to 9. Their indices are [0, 1], so the answer is [0, 1].*

*Similarly, in [3, 2, 4] with target = 6, the numbers 2 and 4 add up to 6. Their indices are [1, 2], so the answer is [1, 2].*

*Our task is to find these two numbers and return their indices.*

***Code:***

*class Solution {*

*public:*

*vector<int> twoSum(vector<int>& nums, int target) {*

*for (int i = 0; i < nums.size(); i++) {*

*for (int j = i + 1; j < nums.size(); j++) {*

*if (nums[i] + nums[j] == target) {*

*return {i, j};*

*}*

*}*

*}*

*return {};*

*}*

*};*

***Question no. 5***

***Leet code Question no. 58***

***Question name: length of Last Word***

***Level: Easy***

***Problem Description:***

*The question asks us to find the length of the last word in a given string. A word is defined as a sequence of non-space characters.*

*For example:*

* *Input: "Hello World" → The last word is "World", and its length is* ***5****.*
* *Input: " fly me to the moon " → The last word is "moon", and its length is* ***4****.*
* *Input: "luffy is still joyboy" → The last word is "joyboy", and its length is* ***6****.*

*Our task is to return the length of the last word in the given string.*

*4o*

***Code:***

class Solution {

public:

    int lengthOfLastWord(string s) {

        int length = 0;

        int i = s.size() - 1;

        while (i >= 0 && s[i] == ' ') {

            i--;

        }

        while (i >= 0 && s[i] != ' ') {

            length++;

            i--;

        }

        return length;

    }

};