**Assignment 9**

1. Add Sum

public class Solution {

    public int[] TwoSum(int[] num, int target) {

        Dictionary<int, int> numIndices = new Dictionary<int, int>();

        for (int i = 0; i < num.Length; i++) {

            int complement = target - num[i];

            if (numIndices.ContainsKey(complement)) {

                return new int[] { numIndices[complement], i };

            }

            numIndices[num[i]] = i;

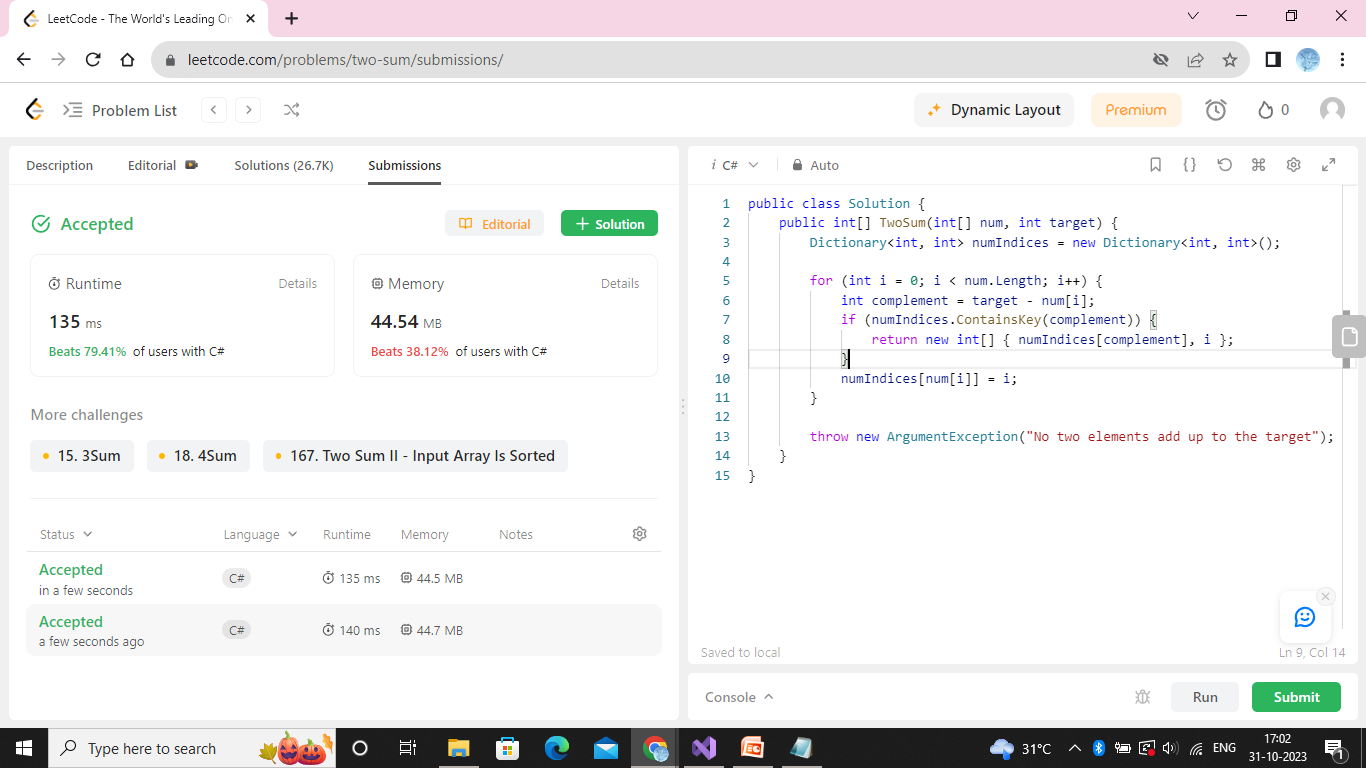
        }

        throw new ArgumentException("No two elements add up to the target");

    }

}

Output



2) Palindrome

public class Solution {

    public bool IsPalindrome(int x) {

        if (x < 0 || (x % 10 == 0 && x != 0)) {

            return false;

        }

        int reversed = 0;

        int original = x;

        while (x > 0) {

            int digit = x % 10;

            reversed = reversed \* 10 + digit;

            x /= 10;

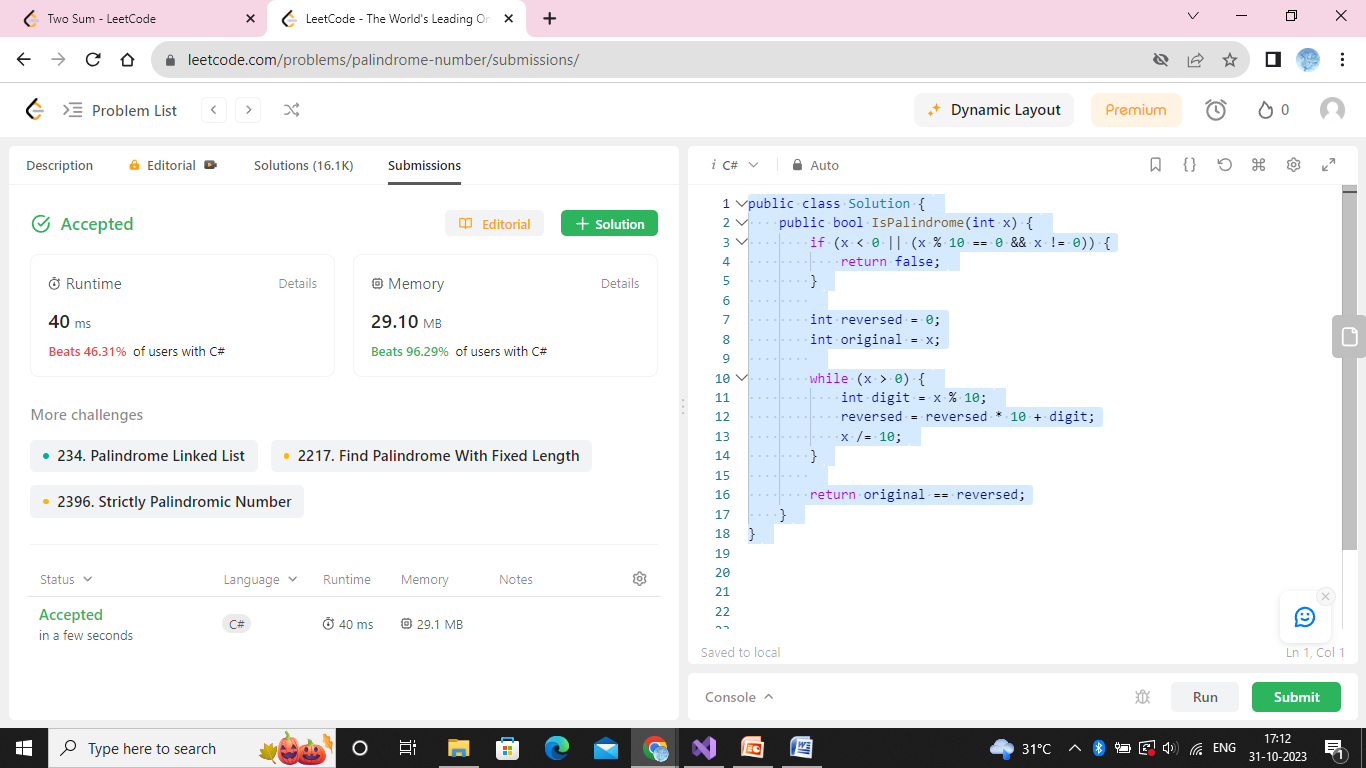
        }

        return original == reversed;

    }

}

Output



1. Remove Duplicates

public class Solution {

    public int RemoveDuplicates(int[] num) {

        if (num.Length == 0)

        {

            return 0;

        }

        int k = 1;

        for (int i = 1; i < num.Length; i++)

        {

            if (num[i] != num[i - 1])

            {

                num[k] = num[i];

                k++;

            }

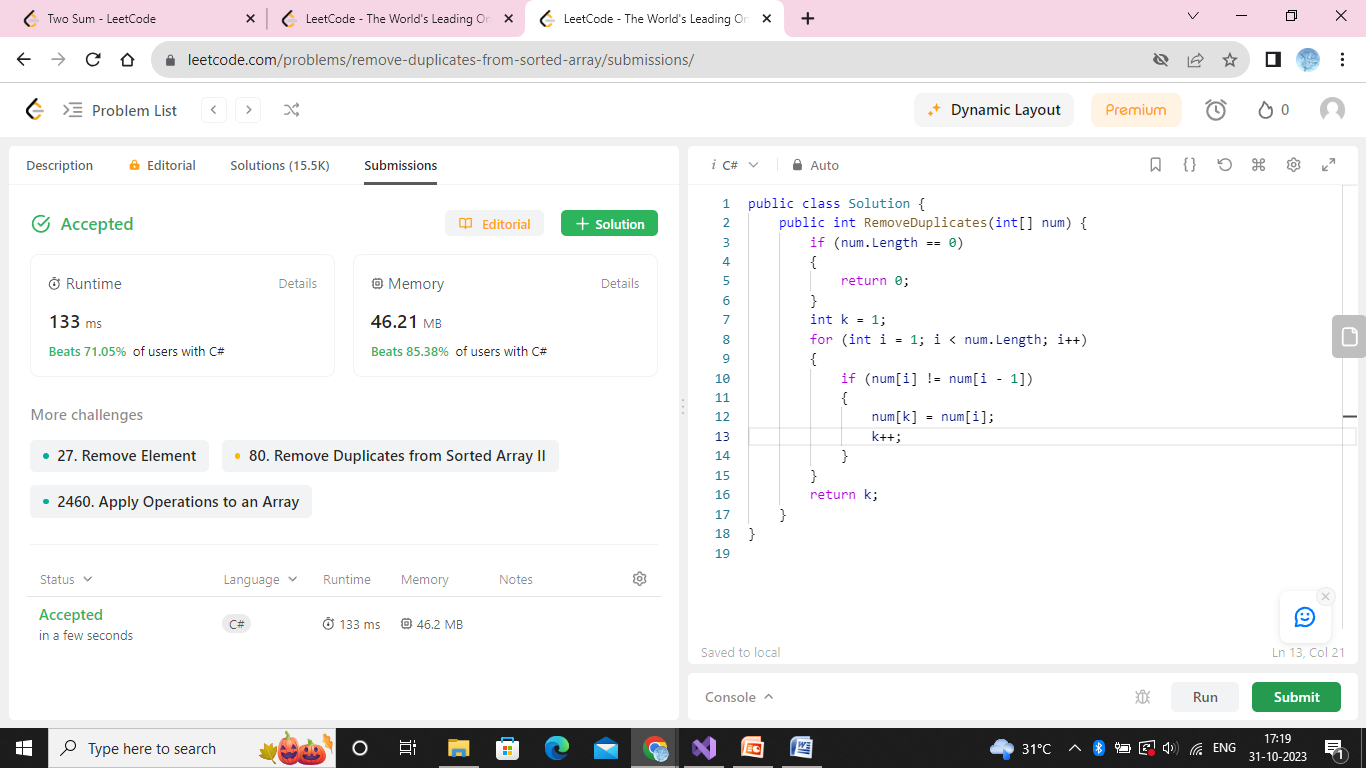
        }

        return k;

    }

}

Output



1. Longest Common Prefix

public class Solution {

    public string LongestCommonPrefix(string[] str) {

        if (str == null || str.Length == 0) {

            return "";

        }

        string prefix = str[0];

        for (int i = 1; i < str.Length; i++) {

            while (str[i].IndexOf(prefix) != 0) {

                prefix = prefix.Substring(0, prefix.Length - 1);

                if (string.IsNullOrEmpty(prefix)) {

                    return "";

                }

            }

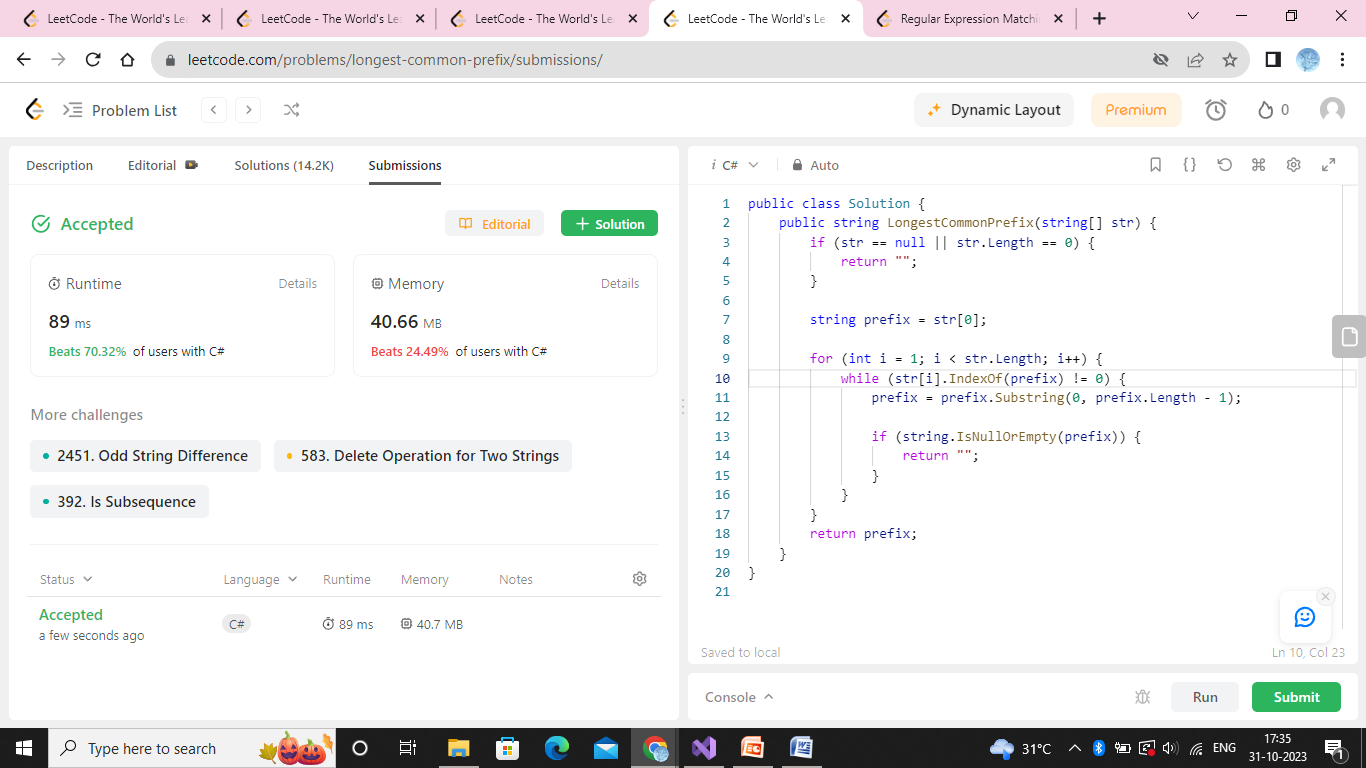
        }

        return prefix;

    }

}

Output



1. Match

public class Solution {

    public bool IsMatch(string s, string p) {

        int m = s.Length;

        int n = p.Length;

        bool[,] dp = new bool[m + 1, n + 1];

        dp[0, 0] = true;

        for (int i = 0; i <= m; i++) {

            for (int j = 1; j <= n; j++) {

                if (p[j - 1] != '\*') {

                    dp[i, j] = i > 0 && (s[i - 1] == p[j - 1] || p[j - 1] == '.') && dp[i - 1, j - 1];

                } else {

                    dp[i, j] = dp[i, j - 2] || (i > 0 && (s[i - 1] == p[j - 2] || p[j - 2] == '.') && dp[i - 1, j]);

                }

            }

        }

        return dp[m, n];

    }

}

Output

