**Logics**

public class Solution {

public int[] TwoSum(int[] nums, int target)

{

int[] result = new int[] {-1,-1};

if(nums == null)

return result;

if(nums.Length < 2)

return result;

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

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

{

int diff = target-nums[i];

if(map.ContainsKey(diff))

{

result[0] = map[diff];

result[1] = i;

return result;

}

if(map.ContainsKey(nums[i]))

continue;

map.Add(nums[i], i);

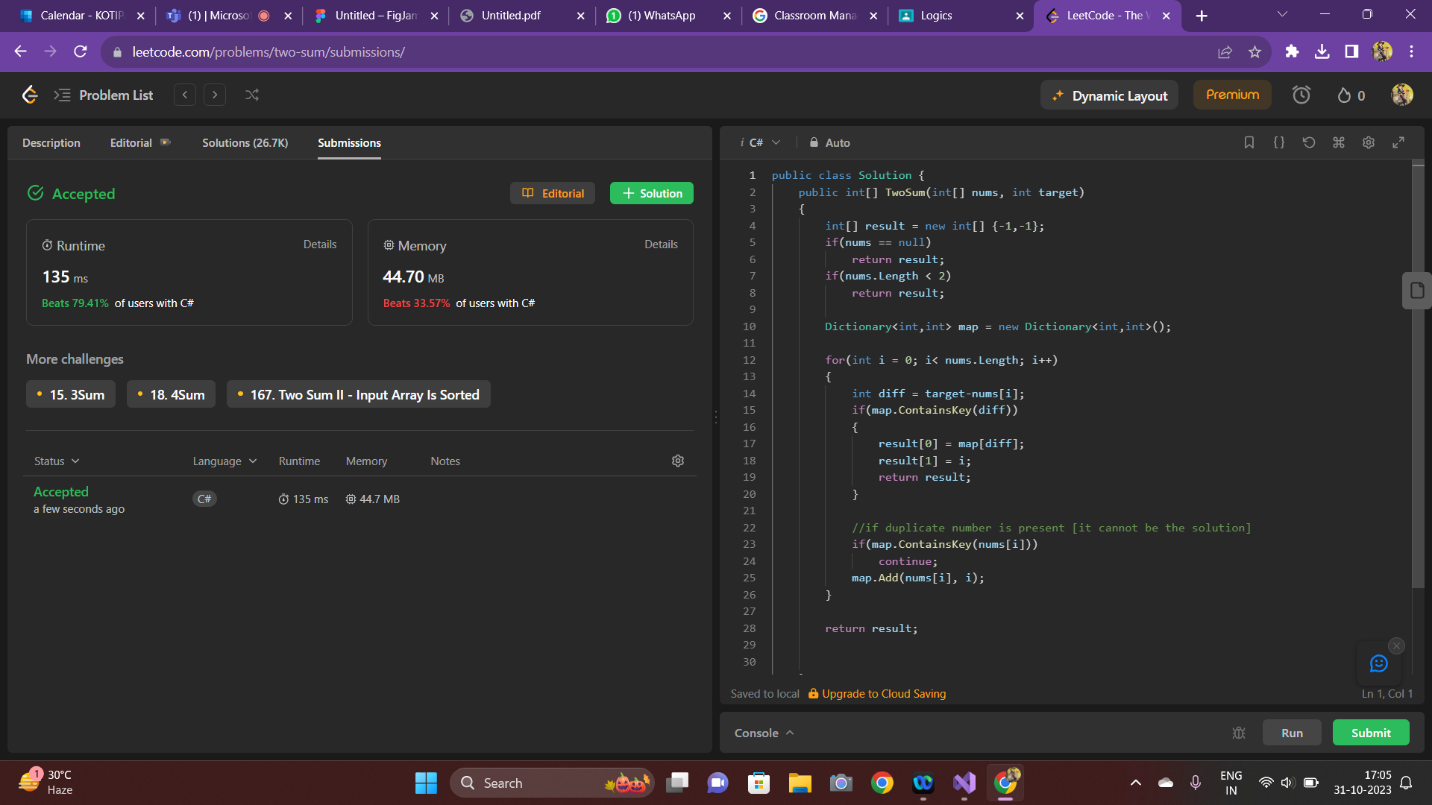
}

return result;

}

}

**Output:**

****

**---2-----**

public class Solution {

public bool IsPalindrome(int x) {

// If it's negative, it's automatically false

if(x < 0){

return false;

}

int remainder = 0;

int reversed = 0;

int original = x;

while (x != 0) {

remainder = x % 10;

reversed = reversed \* 10 + remainder;

x /= 10;

}

if(reversed == original){

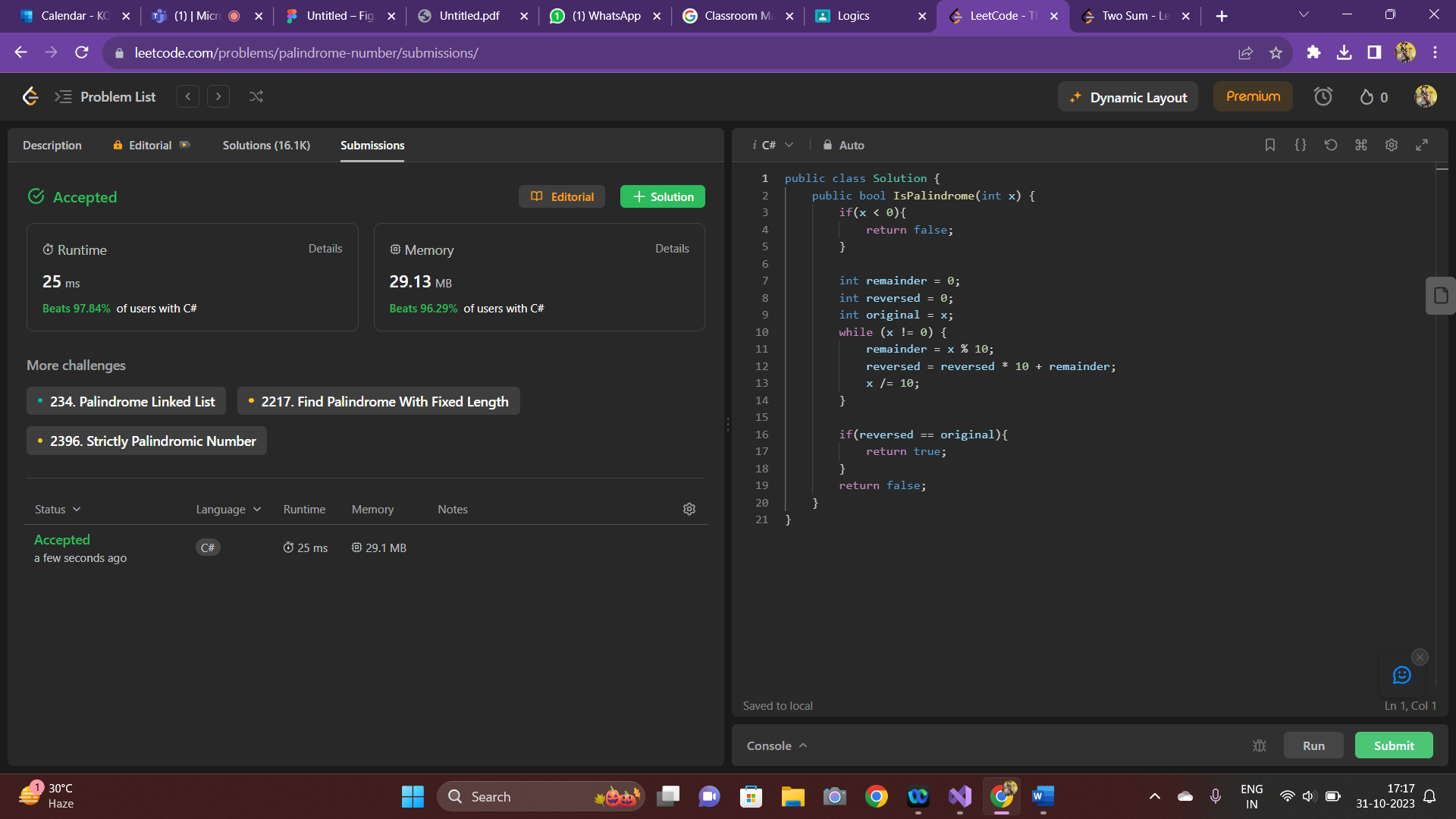
return true;

}

return false;

}

}



**--3—**

public class Solution {

public int RemoveDuplicates(int[] nums) {

if(nums.Count() == 0) return 0;

var iTarget = 0;

for(var i = 0; i < nums.Count(); i++){

if(nums[iTarget] != nums[i]){

nums[++iTarget] = nums[i];

}

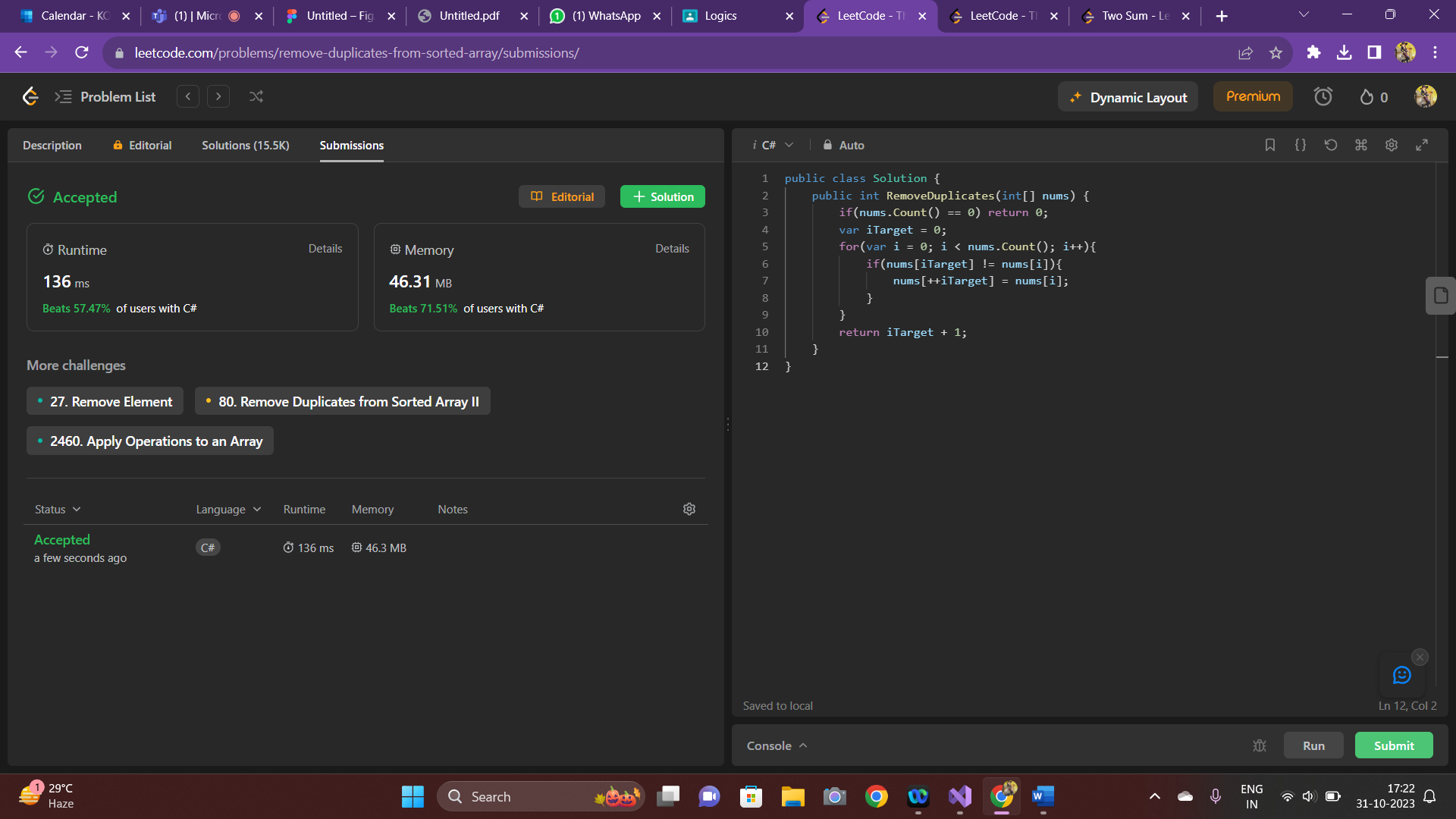
}

return iTarget + 1;

}

}

**Output:**



**--4—**

public class Solution {

public string LongestCommonPrefix(string[] strs) {

int length=strs.Length;

if (length==0) return "";

if (length==1) return strs[0];

int count=strs[0].Length;

string answer=strs[0];

for (int i=1; i<length; i++){

int strsLength=strs[i].Length;

for (int j=0; j<count; j++){

if (j==strsLength||strs[i][j]!=answer[j])

{

answer=answer.Substring(0, j);

count=answer.Length;

if (count==0) return "";

break;

}

}

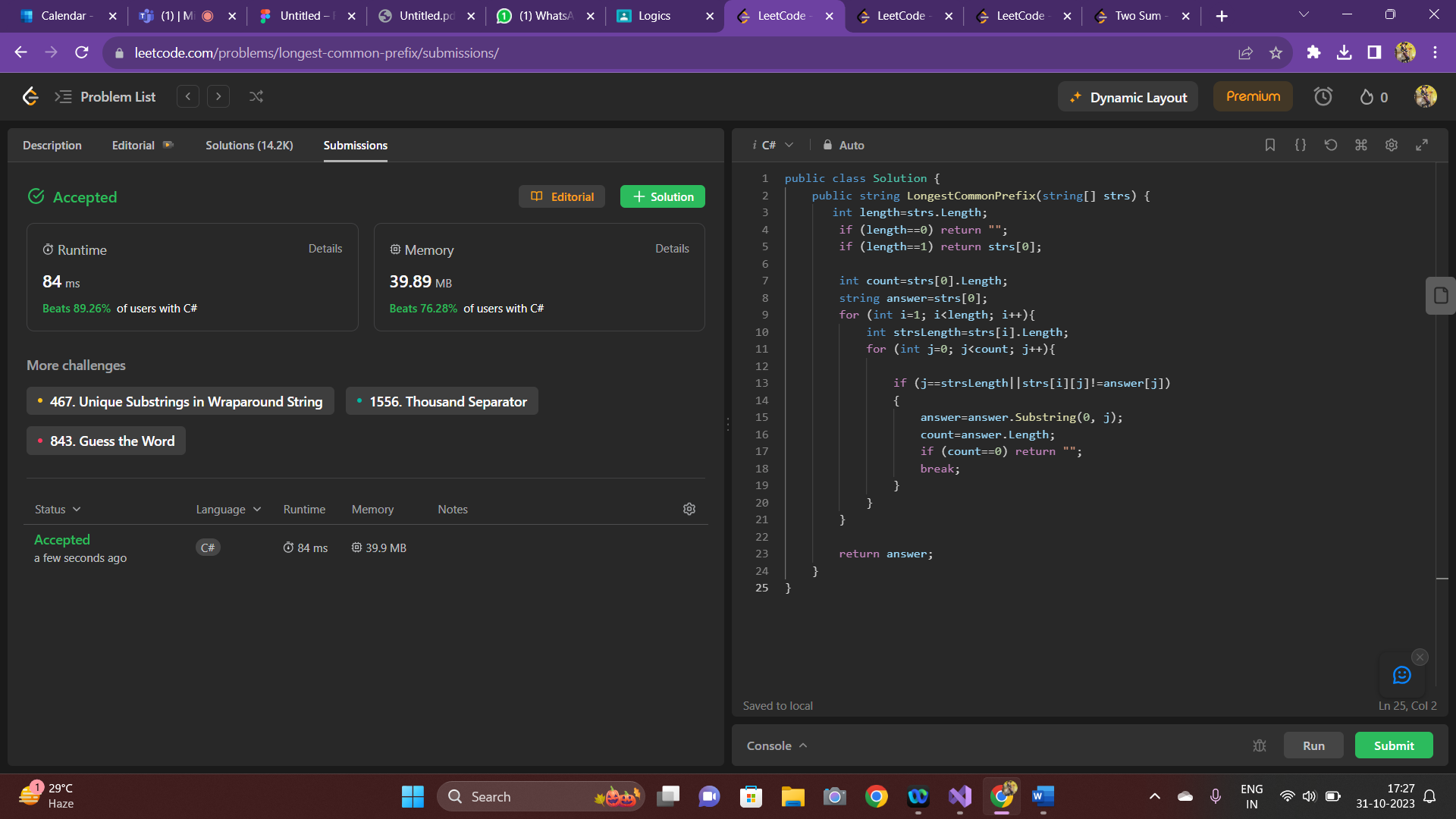
}

return answer;

}

}

**Output:**



**--5—**

public class Solution {

public bool IsMatch(string s, string p) {

if (p.Length == 0)

{

return s.Length == 0;

}

if (p.Length == 1 || p[1] != '\*')

{

if (s.Length == 0 || (p[0] != '.' && p[0] != s[0]))

{

return false;

}

return IsMatch(s.Substring(1), p.Substring(1));

}

int i = -1;

while (i < s.Length && (i < 0 || p[0] == '.' || p[0] == s[i]))

{

if (IsMatch(s.Substring(i + 1), p.Substring(2)))

{

return true;

}

i++;

}

return false;

}

}

**Output:**

