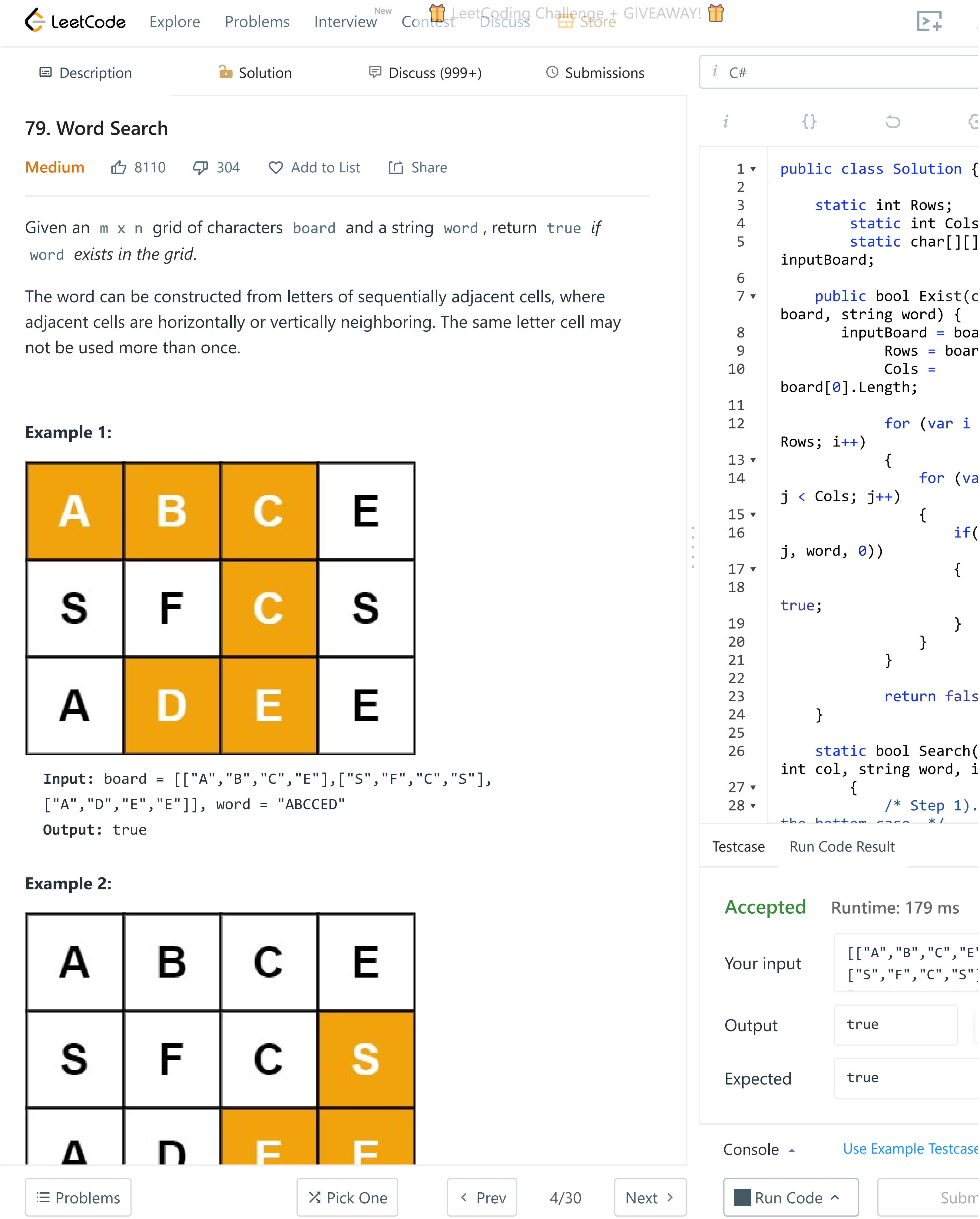

79. Word Search

Given an `m x n` grid of characters `board` and a string `word`, return `true` *if* `word` *exists in the grid*.

The word can be constructed from letters of sequentially adjacent cells, where adjacent cells are horizontally or vertically neighboring. The same letter cell may not be used more than once.

Example 1:

A	B	C	E
S	F	C	S
A	D	E	E

```
Input: board = [["A","B","C","E"],["S","F","C","S"],
["A","D","E","E"]], word = "ABCCED"
Output: true
```

Example 2:

A	B	C	E
S	F	C	S
A	D	E	E

```
public class Solution {

    static int Rows;
        static int Cols
        static char[][]
inputBoard;

    public bool Exist(c
board, string word) {
        inputBoard = boa
            Rows = boar
            Cols =
board[0].Length;

            for (var i
Rows; i++)
            {
                for (va
j < Cols; j++)
                {
                    if(
j, word, 0))
                    {

true;
                    }
                }
            }

            return fals
    }

    static bool Search(
int col, string word, i
        {
            /* Step 1).
the bottom case. */
```

Accepted Runtime: 179 ms

Your input
```
[["A","B","C","E
["S","F","C","S"
```

Output
```
true
```

Expected
```
true
```