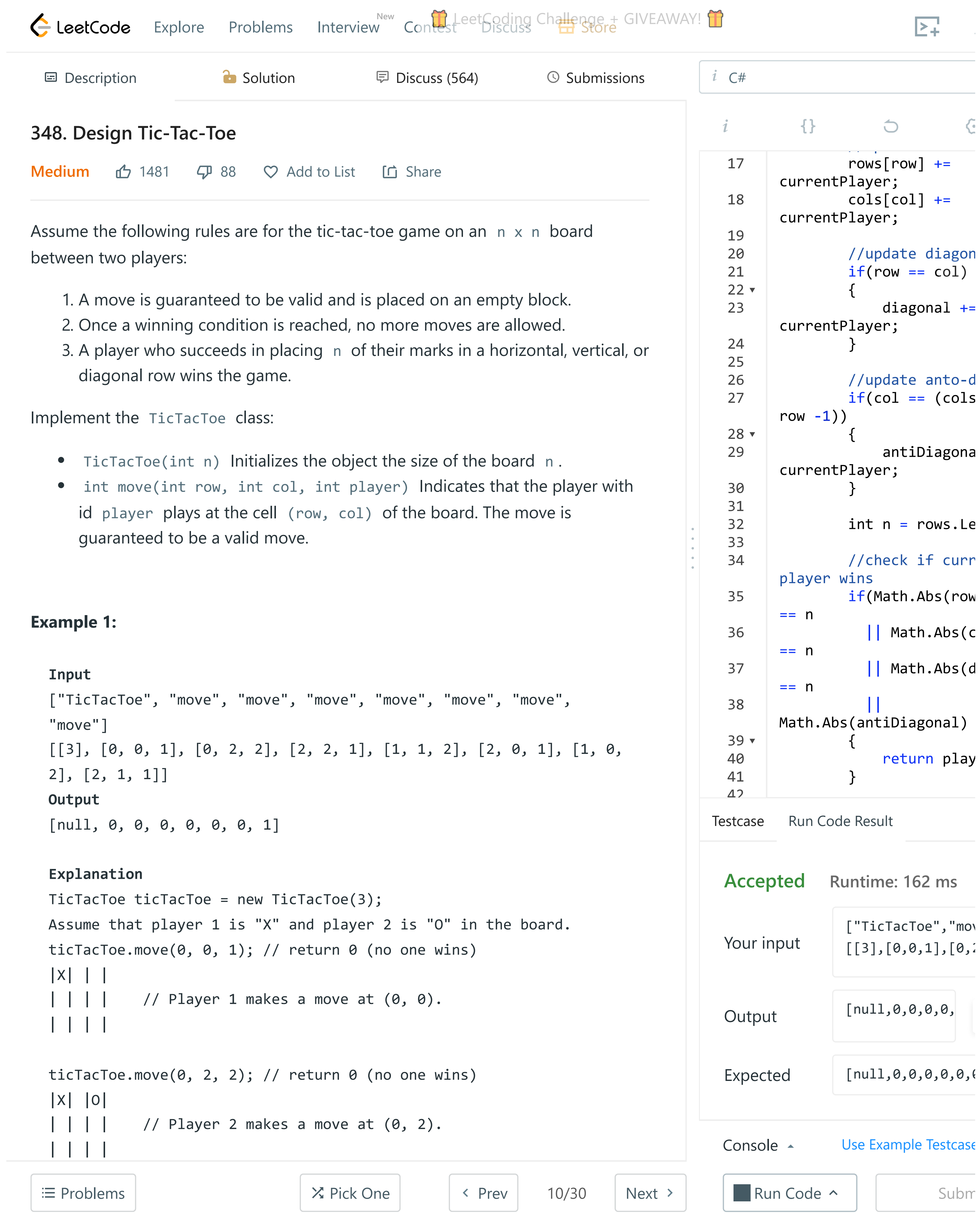

348. Design Tic-Tac-Toe

Medium 1481 88 Add to List Share

Assume the following rules are for the tic-tac-toe game on an `n x n` board between two players:

1. A move is guaranteed to be valid and is placed on an empty block.
2. Once a winning condition is reached, no more moves are allowed.
3. A player who succeeds in placing `n` of their marks in a horizontal, vertical, or diagonal row wins the game.

Implement the `TicTacToe` class:

- `TicTacToe(int n)` Initializes the object the size of the board `n`.
- `int move(int row, int col, int player)` Indicates that the player with id `player` plays at the cell `(row, col)` of the board. The move is guaranteed to be a valid move.

Example 1:

```
Input
["TicTacToe", "move", "move", "move", "move", "move", "move", "move"]
[[3], [0, 0, 1], [0, 2, 2], [2, 2, 1], [1, 1, 2], [2, 0, 1], [1, 0, 2], [2, 1, 1]]
Output
[null, 0, 0, 0, 0, 0, 0, 1]

Explanation
TicTacToe ticTacToe = new TicTacToe(3);
Assume that player 1 is "X" and player 2 is "O" in the board.
ticTacToe.move(0, 0, 1); // return 0 (no one wins)
|X| | |
| | | |    // Player 1 makes a move at (0, 0).
| | | |

ticTacToe.move(0, 2, 2); // return 0 (no one wins)
|X| |O|
| | | |    // Player 2 makes a move at (0, 2).
| | | |
```

```
        rows[row] +=
currentPlayer;
        cols[col] +=
currentPlayer;

        //update diagon
        if(row == col)
        {
            diagonal +=
currentPlayer;
        }

        //update anto-d
        if(col == (cols
row -1))
        {
            antiDiagona
currentPlayer;
        }

        int n = rows.Le

        //check if curr
player wins
        if(Math.Abs(row
== n
           || Math.Abs(c
== n
           || Math.Abs(d
== n
           ||
Math.Abs(antiDiagonal)
        {
            return play
        }
```

Testcase Run Code Result

Accepted Runtime: 162 ms

Your input
```
["TicTacToe","mov
[[3],[0,0,1],[0,2
```

Output
```
[null,0,0,0,0,
```

Expected
```
[null,0,0,0,0,0,0
```