**Name**: Raj K Patel

**Roll No**: 20BCE218

**Course**: 2CSDE93 - Blockchain Technology

**Practical No**: 10

**Aim**: Tick-Tack-Toe in Solidity Programming Language

**Code:**

***// SPDX-License-Identifier: MIT***

pragma solidity ^0.8.0;

contract TicTacToe { address public player1; address public player2;

address public currentPlayer; uint8[3][3] public board; address public winner;

bool public gameFinished = false;

event MoveMade(address indexed player, uint8 row, uint8 col);

event GameFinished(address indexed winner);

constructor(address \_player1, address \_player2) { player1 = \_player1;

player2 = \_player2; currentPlayer = player1;

**}**

modifier onlyPlayers() { require(

player2,

**);**

**\_;**

msg.sender == player1 || msg.sender == "Only players can make a move"

**}**

modifier isGameFinished() { require(!gameFinished, "The game has already

finished");

**\_;**

**}**

function makeMove(uint8 row, uint8 col) public onlyPlayers {

require(

row >= 1 && row <= 3,

"Invalid row. Row must be between 1 and 3"

and 3"

**);**

require(

col >= 1 && col <= 3,

"Invalid column. Column must be between 1

**);**

require(

board[row - 1][col - 1] == 0, "Invalid move. Cell already taken"

**);**

require(msg.sender == currentPlayer, "It's not your turn");

board[row - 1][col - 1] = currentPlayer == player1 ? 1 : 2;

emit MoveMade(currentPlayer, row, col);

***draw***

if (checkWinner(row, col)) { winner = currentPlayer; gameFinished = true;

emit GameFinished(winner);

} else if (boardIsFull()) { gameFinished = true;

**emit GameFinished(address(0)); *// It's a***

} else {

currentPlayer = (currentPlayer == player1)

? player2 : player1;

**}**

**}**

function checkWinner(uint8 row, uint8 col) internal view returns (bool) {

***// Check the row***

if (

board[row - 1][0] == board[row - 1][1] && board[row - 1][0] == board[row - 1][2]

**) {**

return true;

**}**

***// Check the column***

if (

board[0][col - 1] == board[1][col - 1] && board[0][col - 1] == board[2][col - 1]

**) {**

return true;

**}**

***// Check diagonals***

if (

((board[0][0] == board[1][1] && board[0][0]

== board[2][2]) ||

(board[0][2] == board[1][1] &&

board[0][2] == board[2][0]))

**) {**

return true;

**}**

return false;

**}**

function boardIsFull() internal view returns (bool)

**{**

for (uint8 i = 0; i < 3; i++) { for (uint8 j = 0; j < 3; j++) {

if (board[i][j] == 0) { return false;

**}**

**}**

**}**

**return true;**

**}**

**}**







