

Name: Darshil Maru  
Roll No: 20BCE514  
Blockchain Technology

## Practical 10

**Aim: To write a contract code to implement a two-player game Tic-Tac-Toe.**

The image displays two screenshots from a development environment. The top screenshot shows the Solidity Compiler interface with the following details:

- COMPILER:** 0.4.24+commit.e67f0147
- LANGUAGE:** Solidity
- EVM VERSION:** compiler default
- COMPILER CONFIGURATION:** Auto compile, Enable optimization (200), Hide warnings
- CONTRACT:** TicTacToe [TicTacToe.sol]
- Code:**

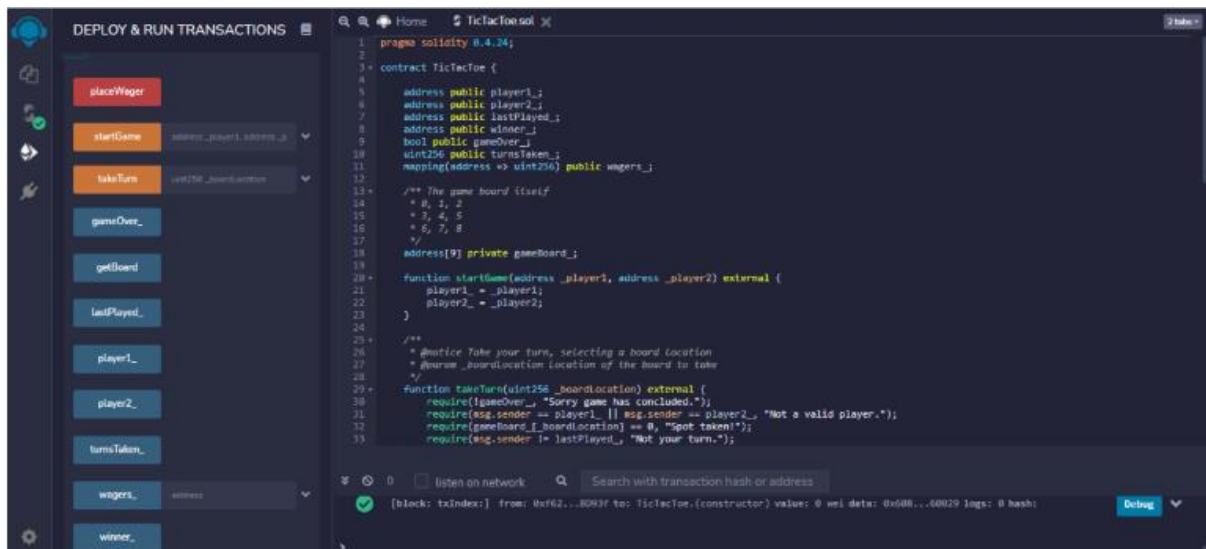
```
1 pragma solidity 0.4.24;
2
3 contract TicTacToe {
4
5     address public player1;
6     address public player2;
7     address public lastPlayed;
8     address public winner;
9     bool public gameOver;
10    uint256 public turnsTaken;
11    mapping(address => uint256) public wagers;
12
13    /** The game board itself
14     * 0, 1, 2
15     * 3, 4, 5
16     * 6, 7, 8
17     */
18    address[9] private gameBoard;
19
20    function startGame(address _player1, address _player2) external {
21        player1 = _player1;
22        player2 = _player2;
23    }
24
25    /**
26     * @notice Take your turn, selecting a board location
27     * @param _boardLocation Location of the board to take
28     */
29    function takeTurn(uint256 _boardLocation) external {
30        require(!gameOver, "Sorry game has concluded.");
31        require(msg.sender == player1 || msg.sender == player2, "Not a valid player.");
32        require(gameBoard[_boardLocation] == 0, "Spot taken!");
33        require(msg.sender != lastPlayed, "Not your turn.");
```
- Transaction:** [call] from: 0xf624701432e5452c01c8992d0a4231f6588093f to: TicTacToe-player2() data: 0xc0b...ecf5f

The bottom screenshot shows the 'DEPLOY & RUN TRANSACTIONS' interface with the following details:

- ENVIRONMENT:** Injected Web3
- ACCOUNT:** 0xf62...BD93f (0.50421461)
- GAS LIMIT:** 3000000
- VALUE:** 0 Wei
- CONTRACT:** TicTacToe - TicTacToe.sol
- Buttons:** Deploy, Publish to IPFS, OR, At Address, Load contract from Address
- Transactions recorded:** 1
- Deployed Contracts:** 1
- Transaction:** creation of TicTacToe pending...

On the right, a MetaMask Notification window is open, showing the contract deployment details:

- Account:** Account1
- Network:** Ropsten Test Network
- URL:** https://remix.ethereum.org
- Contract:** CONTRACT DEPLOYMENT
- Details:** Estimated gas fee: 0.002506 ETH, Total: 0.002506 ETH
- Buttons:** Reject, Confirm



## Smart Contract:

```

pragma solidity 0.4.24;

contract TicTacToe {

    address public player1_;
    address public player2_;
    address public lastPlayed_;
    address public winner_;
    bool public gameOver_;
    uint256 public turnsTaken_;
    mapping(address => uint256) public wagers_;

    /** The game board itself
     * 0, 1, 2
     * 3, 4, 5
     * 6, 7, 8
     */
    address[9] private gameBoard_;

    function startGame(address _player1, address _player2) external {
        player1_ = _player1;
        player2_ = _player2;
    }

    /**
     * @notice Take your turn, selecting a board location
     * @param _boardLocation Location of the board to take
     */
    function takeTurn(uint256 _boardLocation) external {
        require(!gameOver_, "Sorry game has concluded.");
        require(msg.sender == player1_ || msg.sender == player2_, "Not a valid player.");
        require(gameBoard_[_boardLocation] == 0, "Spot taken!");
        require(msg.sender != lastPlayed_, "Not your turn.");
    }


```

```

    gameBoard_[_boardLocation] = msg.sender;
    lastPlayed_ = msg.sender;
    turnsTaken_++;

    if (isWinner(msg.sender)) {
        winner_ = msg.sender;
        gameOver_ = true;
        msg.sender.transfer(address(this).balance);
    } else if (turnsTaken_ == 9) {
        gameOver_ = true;
        player1_.transfer(wagers_[player1_]);
        player2_.transfer(wagers_[player2_]);
    }
}

/**
 * Winning filters:
 * 0, 1, 2
 * 3, 4, 5
 * 6, 7, 8
 *
 * 3 in a row:
 * [0,1,2] || [3,4,5] || [6,7,8]
 *
 * 3 in a column:
 * [0,3,6] || [1,4,7] || [2,5,8]
 *
 * Diagonals:
 * [0,4,8] || [6,4,2]
 */
function isWinner(address player) private view returns(bool) {
    uint8[3][8] memory winningFilters = [

```

```

        [0,1,2],[3,4,5],[6,7,8], // rows
        [0,3,6],[1,4,7],[2,5,8], // columns
        [0,4,8],[6,4,2] // diagonals
    ];

    for (uint8 i = 0; i < winningFilters.length; i++) {
        uint8[3] memory filter = winningFilters[i];
        if (
            gameBoard_[filter[0]]==player &&
            gameBoard_[filter[1]]==player &&
            gameBoard_[filter[2]]==player
        ) {
            return true;
        }
    }
}

function placeWager() external payable {
    require(msg.sender == player1_ || msg.sender == player2_, "Not a valid player.");
    wagers_[msg.sender] = msg.value;
}

function getBoard() external view returns(address[9]) {
    return gameBoard_;
}
}

```