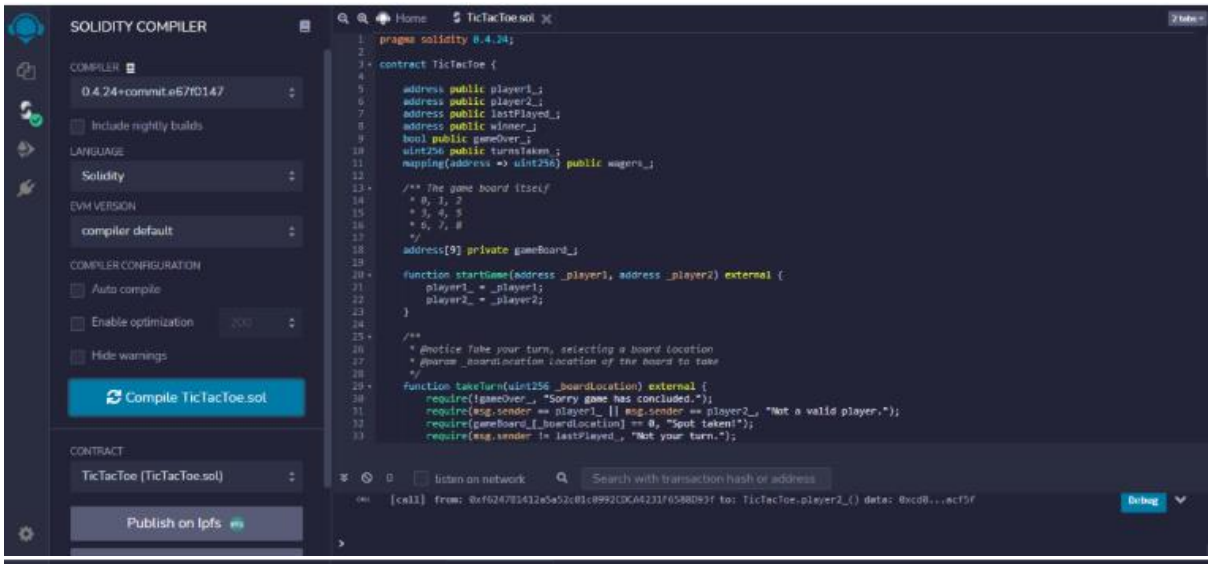
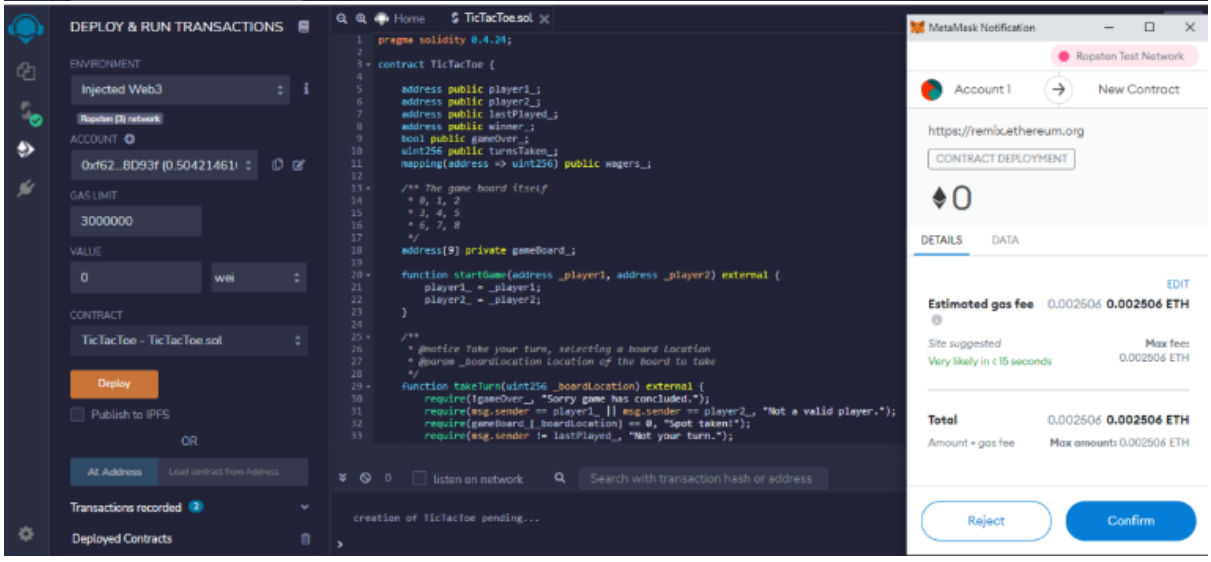
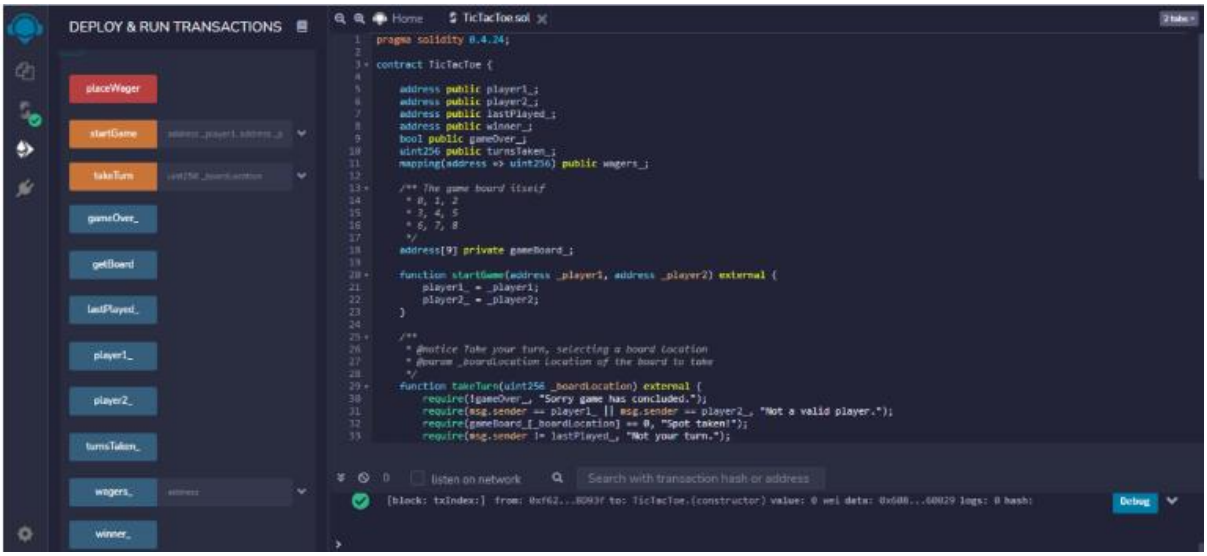
Name: Darshil Maru
Roll No: 20BCE514
Blockchain Technology

Practical 10

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







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_;
    }
}
```