

PRACTICAL - 10

To write a contract code to implement a two-player game (with a wager on the line) of Tic-Tac-Toe, also known as Noughts and Crosses: (Ethereum programming)

2CSDE93

18BCE106
DIVYA MAHUR

CODE:

```
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_;
    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_;  
}  
}
```

OUTPUT SCREENSHOTS:

The screenshot shows the Solidity Compiler interface in the Remix IDE. The code editor displays the `TicTacToe` contract with the `startGame` function. The function takes two parameters: `_player1` and `_player2`. It initializes the board locations for both players and sets the `winner` variable to `null`.

```
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_;
    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);
    }
}
```

The screenshot shows the Solidity Compiler interface in the Remix IDE. The code editor displays the `TicTacToe` contract with the `takeTurn` function. This function checks if the game is over or if the sender is a valid player. It then updates the board location, marks the sender as the last player, increments the turn counter, and checks if the sender is a winner. If the sender is a winner, it sets the `winner` variable and marks the game as over, transferring the balance to the sender.

```
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_;
    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);
    }
}
```

SOLIDITY COMPILER

COMPILER: 0.4.24+commit.e67f0147

LANGUAGE: Solidity

EVM VERSION: compiler default

COMPILE CONFIGURATION: Auto compile, Enable optimization (200), Hide warnings

Contract: TicTacToe (Prac10.sol)

Compile Prac10.sol

Compilation Details: ABI, Bytecode

```

36 * @notice Take your turn, selecting a board location
37 * @param _boardLocation Location of the board to take
38 */
39 function takeTurn(uint256 _boardLocation) external {
40     require(!_gameOver_, "Sorry game has concluded.");
41     require(msg.sender == player1_ || msg.sender == player2_, "Not a valid player.");
42     require(gameBoard_[_boardLocation] == 0, "Spot taken!");
43     require(msg.sender != lastPlayed_, "Not your turn.");
44 }
45
46
47
48
49
50
51
52
53
54

```

- Welcome to Remix 0.19.0 -

You can use this terminal to:

- Check transactions details and start debugging.
- Execute JavaScript scripts:
 - Input a script directly in the command line interface
 - Select a Javascript file in the file explorer and then run `remix.execute()` or `remix.exeCurrent()` in the command line interface
 - Right click on a JavaScript file in the file explorer and then click 'Run'

The following libraries are accessible:

- web3 version 1.5.2
- ethers.js
- remix (run remix.help() for more info)

creation of TicTacToe pending...

[vm] from: 0x5B3...eddC4 to: TicTacToe.(constructor) value: 0 wei data: 0x608...40029 logs: 0 hash: 0x87f...0c5e4

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: JavaScript VM (London)

ACCOUNT: 0x5B3...eddC4 (99.999999ε)

GAS LIMIT: 3000000

VALUE: 0 wei

CONTRACT: TicTacToe - Prac10.sol

Deploy

Publish to IPFS

OR

At Address: Load contract from Address

Transactions recorded: 1

Deployed Contracts: TICTACTOE AT 0XD91...39138 (MEMORY)

```

36 * @notice Take your turn, selecting a board location
37 * @param _boardLocation Location of the board to take
38 */
39 function takeTurn(uint256 _boardLocation) external {
40     require(!_gameOver_, "Sorry game has concluded.");
41     require(msg.sender == player1_ || msg.sender == player2_, "Not a valid player.");
42     require(gameBoard_[_boardLocation] == 0, "Spot taken!");
43     require(msg.sender != lastPlayed_, "Not your turn.");
44 }
45
46
47
48
49
50
51
52
53
54

```

- Welcome to Remix 0.19.0 -

You can use this terminal to:

- Check transactions details and start debugging.
- Execute JavaScript scripts:
 - Input a script directly in the command line interface
 - Select a Javascript file in the file explorer and then run `remix.execute()` or `remix.exeCurrent()` in the command line interface
 - Right click on a JavaScript file in the file explorer and then click 'Run'

The following libraries are accessible:

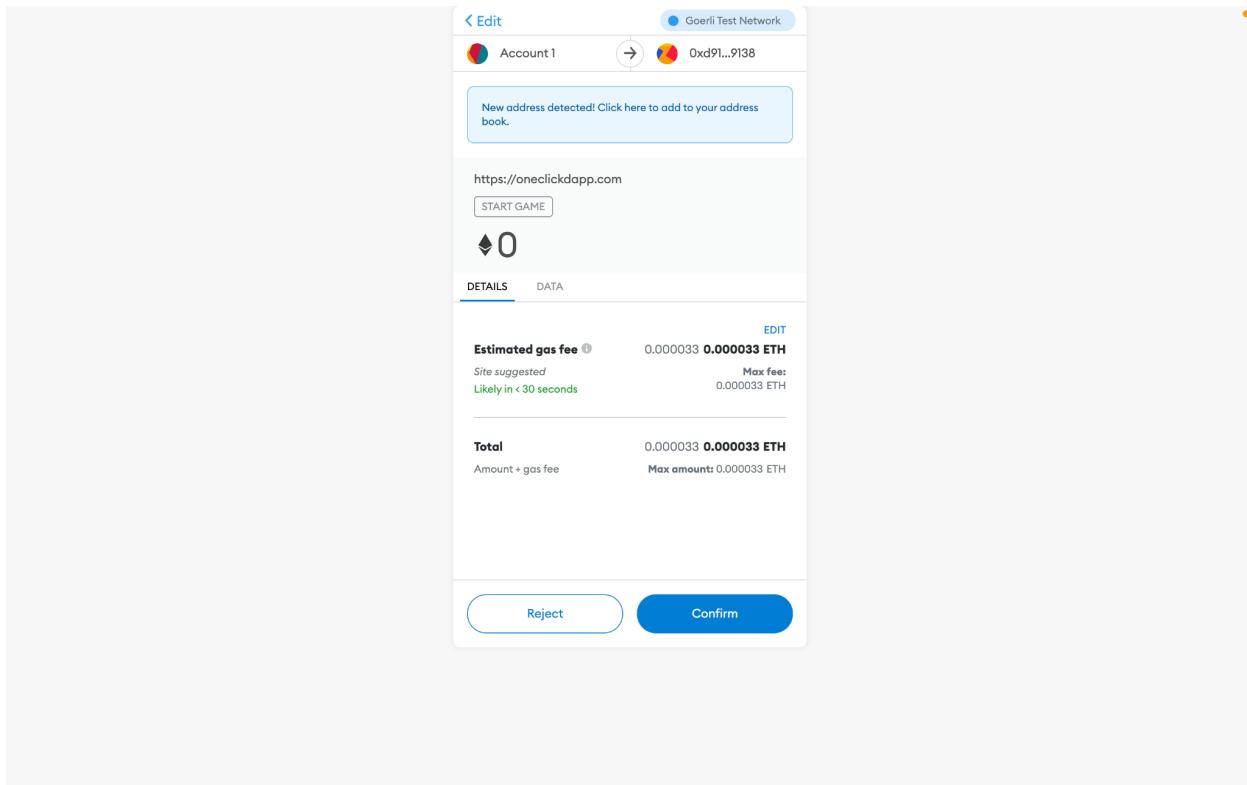
- web3 version 1.5.2
- ethers.js
- remix (run remix.help() for more info)

creation of TicTacToe pending...

[vm] from: 0x5B3...eddC4 to: TicTacToe.(constructor) value: 0 wei data: 0x608...40029 logs: 0 hash: 0x87f...0c5e4

The screenshot shows a web browser window with the URL oneclickdapp.com/shelter-david. The page title is "Tic Tac Toe". It displays a "Contract" section with the address 0xd9145CCE52D386f254917e481eB44e9943F39138 and an "ABI" section showing function signatures like `takeTurn`, `getBoard`, `gameOver_`, etc. Below these are "Read" and "Write" buttons. The "Write" button is active, showing a form for the `startGame` function with two input fields: `_PLAYER1` (containing "i.e. 0x261b45d85ccfeabb11f022eba346ee8d1cd488c0") and `_PLAYER2` (containing "i.e. 0x261b45d85ccfeabb11f022eba346ee8d1cd488c0"). A "SUBMIT" button is at the bottom of the form.

This screenshot shows the same dapp interface after a transaction has been submitted. The "startGame" form now has a different set of player addresses: `_PLAYER1` is "0xb5a9E2e177b51B69d4A4ef3f641186Bf9EdD7C7" and `PLAYER2` is "0x5ed7C44c6675bb8589e7cd8B94d7f9AC854C5C3d". The "SUBMIT" button is still present.



oneclickdapp.com/shelter-david

← → ⌛ 🔍 0 ⓘ

Contract: 0xd9145CCE52D386f254917e481eB44e9943F39138

ABI: takeTurn, getBoard, gameOver_, lastPlayed_, player1_, turnsTaken_, placeWager, winner_, wagers_, pla...

Read Write

placeWager
startGame (address,address)
takeTurn (uint256)

startGame (address,address)

Success!

©2021 Patrick Gallagher

Screenshot of a web browser showing the OneClickDapp interface for a Tic Tac Toe Dapp.

The browser address bar shows: `oneclickdapp.com/shelter-david`.

The main interface displays the **Tic Tac Toe** application. It includes:

- Contract:** Address: `0xd9145CCE52D386f254917e481eB44e9943F39138`
- ABI:** Shows function signatures: `takeTurn, getBoard, gameOver_, lastPlayed_, player1_, turnsTaken_, placeWager, winner_, wagers_, pla...`
- Buttons:** `Read` (disabled), `Write`
- Function Call:** `takeTurn (uint256)` with input field `BOARDLOCATION` containing `5`, and a `SUBMIT` button.
- Bottom Footer:** ©2021 Patrick Gallagher
- Gas Fee Confirmation Overlay:** Shows Goerli Test Network, Account 1 (`0xd91...9138`), Estimated gas fee: `0.000032 0.000032 ETH`, Total: `0.000032 0.000032 ETH`, and buttons `Reject` and `Confirm`. A message indicates a new address was detected.
- Right Sidebar:** Shows a confirmation message: `Confirmed transaction Transaction 6 confirmed! on Etherscan`.

Tic Tac Toe

(no network) Created 17 November 2021

Contract 0xd9145CCE52D386f254917e481eB44e9943F39138

ABI takeTurn, getBoard, gameOver_, lastPlayed_, player1_, turnsTaken_, placeWager, winner_, wagers_, pla...

Read **Write**

placeWager
startGame (address,address)
takeTurn (uint256)

takeTurn (uint256)

Success!

©2021 Patrick Gallagher

Tic Tac Toe

(no network) Created 17 November 2021

Contract 0xd9145CCE52D386f254917e481eB44e9943F39138

ABI takeTurn, getBoard, gameOver_, lastPlayed_, player1_, turnsTaken_, placeWager, winner_, wagers_, pla...

Read **Write**

gameOver_
getBoard
lastPlayed_
player1_
player2_
turnsTaken_
wagers_ (address)
winner_

takeTurn (uint256)

Success!

©2021 Patrick Gallagher

