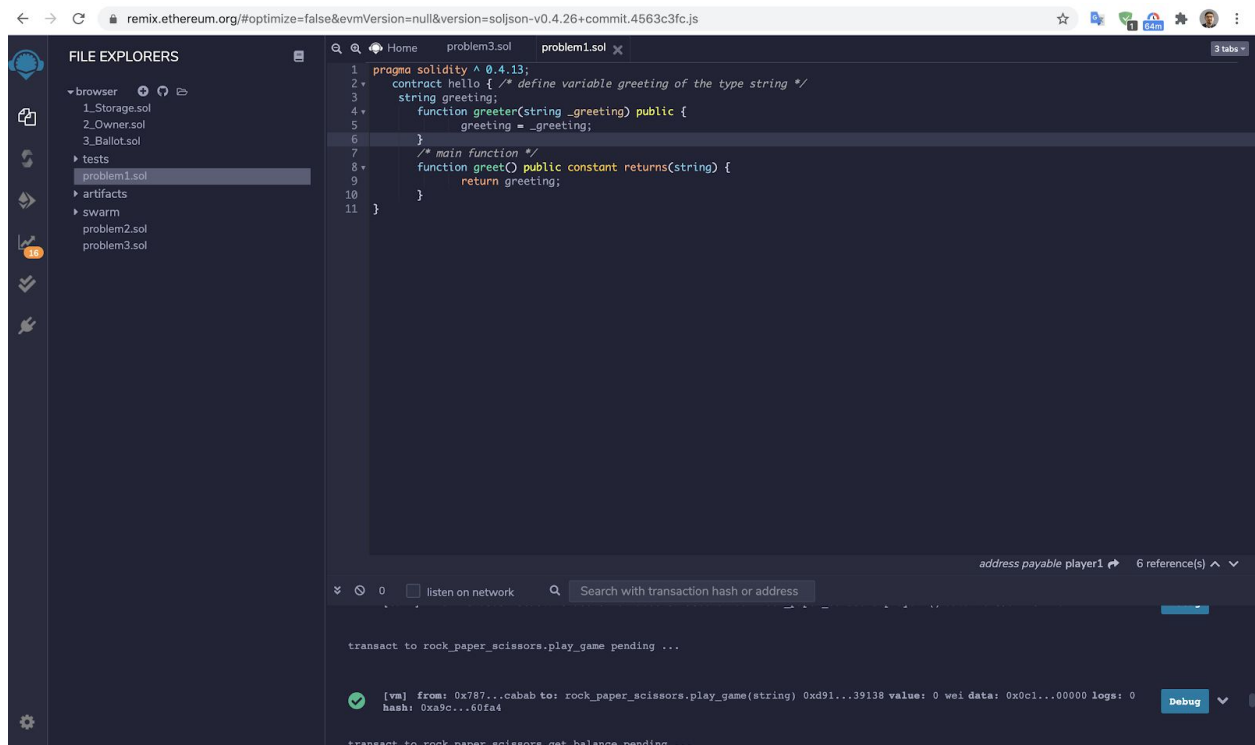
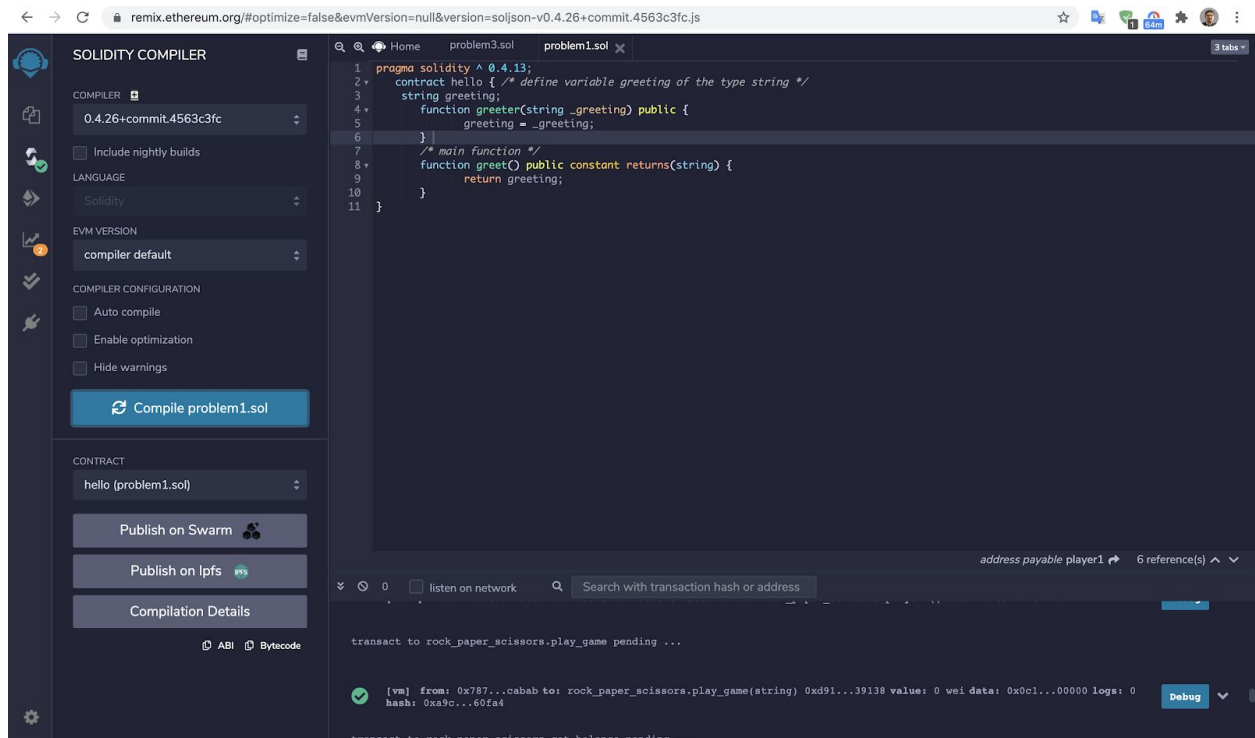


Problem 1

Part -1) Code



Part 2) Compile



Part 3) Deploy and output

remix.ethereum.org/#optimize=false&evmVersion=null&version=soljson-v0.4.26+commit.4563c3fc.js

DEPLOY & RUN TRANSACTIONS

0x787...cabaB (104.999999)

GAS LIMIT: 3000000

VALUE: 0 ether

CONTRACT: hello - browser/problem1.sol

Deploy

☐ Publish to IPFS

OR

At Address: Load contract from Address

Transactions recorded: 2

Deployed Contracts

HELLO AT 0XC58...905F2 (MEMORY)

greeter: Dhruvil

greet

0: string: Dhruvil

Low level interactions

CALLDATA

Transact

```
1 pragma solidity ^0.4.13;
2 contract hello { /* define variable greeting of the type string */
3   string greeting;
4   function greeter(string _greeting) public {
5     greeting = _greeting;
6   }
7   /* main function */
8   function greet() public constant returns(string) {
9     return greeting;
10  }
11 }
```

address payable player1 6 reference(s)

listen on network Search with transaction hash or address

transact to rock_paper_scissors.play_game pending ...

[vm] from: 0x787...cabaB to: rock_paper_scissors.play_game(string) 0xd91...39138 value: 0 wei data: 0x0c1...00000 logs: 0 hash: 0xa9c...60fa4 Debug

transact to rock_paper_scissors.get_balance pending ...

Problem 2):

Part 1) Code

remix.ethereum.org/#optimize=false&evmVersion=null&version=soljson-v0.4.26+commit.4563c3fc.js

FILE EXPLORERS

browser

- 1.Storage.sol
- 2.Owners.sol
- 3.Ballots.sol

tests

problem1.sol

artifacts

- hello_metadata.json
- hello.json
- rock_paper_scissors_metadata.json
- rock_paper_scissors.json
- find_max_metadata.json
- find_max.json

swarm

- problem2_sol
- problem3.sol

```
1 pragma solidity ^0.4.13;
2 contract find_max { /* define variable greeting of the type string */
3
4   function my_max(int var1,int var2) public pure returns(int) {
5     return var1 >= var2 ? var1 : var2;
6   }
7 }
```

ContractDefinition find_max 0 reference(s)

listen on network Search with transaction hash or address

transact to rock_paper_scissors.play_game pending ...

[vm] from: 0x787...cabaB to: rock_paper_scissors.play_game(string) 0xd91...39138 value: 0 wei data: 0x0c1...00000 logs: 0 hash: 0xa9c...60fa4 Debug

transact to rock_paper_scissors.get_balance pending ...

Part 2) Compile

The screenshot shows the Remix Ethereum IDE interface. On the left, the 'SOLIDITY COMPILER' panel is active, displaying the compiler version '0.4.26+commit.4563c3fc' and various configuration options. The main editor shows a Solidity file named 'problem2.sol' with the following code:

```
1 pragma solidity ^0.4.13;
2 contract find_max { /* define variable greeting of the type string */
3
4     function my_max(int var1,int var2) public pure returns(int) {
5         return var1 >= var2 ? var1 : var2;
6     }
7 }
```

The console at the bottom shows a successful compilation and a transaction to 'rock_paper_scissors.play_game' with a value of 0 wei and data '0x0c1...00000'. The transaction is pending.

Part 3) Deploy and test

The screenshot shows the 'DEPLOY & RUN TRANSACTIONS' panel in the Remix Ethereum IDE. The environment is set to 'JavaScript VM'. The account '0x787...cabaB' is selected. The gas limit is 3000000. The value is 0 ether. The contract 'find_max' is selected. The 'Deploy' button is highlighted. The console shows a successful deployment and a transaction to 'rock_paper_scissors.play_game' with a value of 0 wei and data '0x0c1...00000'. The transaction is pending.

Problem 3 :

Part 1) Solidity Program:

```
pragma solidity ^ 0.5.11;
```

```
contract rock_paper_scissors {  
    string public player1_choice;  
    string public player2_choice;  
    address payable public player1;  
    address payable public player2;  
    mapping (string => mapping(string => int)) winner_in_game;
```

```
    constructor() public payable{  
        winner_in_game["rock"]["rock"] = 0;  
        winner_in_game["rock"]["paper"] = 2;  
        winner_in_game["rock"]["scissors"] = 1;  
        winner_in_game["paper"]["rock"] = 1;  
        winner_in_game["paper"]["paper"] = 0;  
        winner_in_game["paper"]["scissors"] = 2;  
        winner_in_game["scissors"]["rock"] = 2;  
        winner_in_game["scissors"]["paper"] = 1;  
        winner_in_game["scissors"]["scissors"] = 0;  
    }  
}
```

```
    modifier verify_enough_cash(uint amount){  
        if (msg.value<amount)  
            revert();  
        else  
            _;  
    }  
}
```

```
    modifier not_already_registered() {  
        if(player1==msg.sender || player2==msg.sender){  
            revert();  
        } else {  
            _;  
        }  
    }  
}
```

```
    function register_player() public payable verify_enough_cash(5) not_already_registered() {  
        if(uint(player1)==0){  
            player1 = msg.sender;  
        } else if (uint(player2) == 0) {  
            player2 = msg.sender;  
        }  
    }  
}
```

```

    }
}

function play_game(string memory choice) public payable returns (int w) {
    if(msg.sender==player1){
        player1_choice = choice;
    } else if(msg.sender==player2){
        player2_choice = choice;
    }

    if(bytes(player1_choice).length!=0 && bytes(player2_choice).length!=0){
        int winner = winner_in_game[player1_choice][player2_choice];
        if(winner==1){
            player1.send(address(this).balance);
        } else if (winner==2){
            player2.send(address(this).balance);
        } else {
            player1.send(address(this).balance/2);
            //since we already send half amount to player1 we will send remaining amount to
player2
            player2.send(address(this).balance);
        }
        return winner;
    } else {
        return -1;
    }
}

function prize_amount() public returns (uint x){
    return address(this).balance;
}

function get_balance() public returns (uint x){
    return msg.sender.balance;
}

}

```

Part 2) Compile the program:

remix.ethereum.org/#optimize=false&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js

SOLIDITY COMPILER

COMPILER: 0.5.17+commit.d19bba13

LANGUAGE: Solidity

EVM VERSION: compiler default

COMPILER CONFIGURATION

- ☐ Auto compile
- ☐ Enable optimization
- ☐ Hide warnings

[Compile problem3.sol](#)

CONTRACT: rock_paper_scissors (problem3.sol)

[Publish on Swarm](#)

[Publish on Ipfs](#)

[Compilation Details](#)

[ABI](#) [Bytecode](#)

Warning: Failure condition of 'send' ignored. Consider using 'transfer' instead.

problem3.sol

```

1 pragma solidity ^0.5.11;
2 contract rock_paper_scissors {
3     string public player1_choice;
4     string public player2_choice;
5     address payable public player1;
6     address payable public player2;
7     mapping (string => mapping(string => int)) winner_in_game;
8
9     constructor() public payable{
10         winner_in_game["rock"]["rock"] = 0;
11         winner_in_game["rock"]["paper"] = 2;
12         winner_in_game["rock"]["scissors"] = 1;
13         winner_in_game["paper"]["rock"] = 1;
14         winner_in_game["paper"]["paper"] = 0;
15         winner_in_game["paper"]["scissors"] = 2;
16         winner_in_game["scissors"]["rock"] = 2;
17         winner_in_game["scissors"]["paper"] = 1;
18         winner_in_game["scissors"]["scissors"] = 0;
19     }
20
21     modifier verify_enough_cash(uint amount){
22         if (msg.value < amount)
23             revert();
24         _;
25     }
26
27     modifier not_already_registered() {
28         if (player1 == msg.sender || player2 == msg.sender){
29             revert();
30         } else {
31             _;
32         }
33     }
34 }
35
36 function play_game(string memory _player1, string memory _player2) public not_already_registered() {
37     player1 = payable(msg.sender);
38     player2 = payable(msg.sender);
39     player1_choice = _player1;
40     player2_choice = _player2;
41     winner_in_game[player1_choice][player2_choice]++;
42 }

```

[address payable player1](#) 6 reference(s)

transact to rock_paper_scissors.play_game pending ...

[vm] from: 0x787...cabab to: rock_paper_scissors.play_game(string) 0xd91...39138 value: 0 wei data: 0x0c1...00000 logs: 0
hash: 0xa9c...60fa4 [Debug](#)

transact to rock_paper_scissors.get_balance pending ...

Part 3) Deploy the program :

remix.ethereum.org/#optimize=false&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: JavaScript VM

ACCOUNT: 0x787...cabab (104.999999 ether)

GAS LIMIT: 3000000

VALUE: 0 ether

CONTRACT: rock_paper_scissors - browser/probl

[Deploy](#)

☐ Publish to IPFS

OR

[At Address](#) [Load contract from Address](#)

Transactions recorded: 2

Deployed Contracts

ROCK_PAPER_SCISSORS AT 0XD91...39138 (MEMORY)

[get_balance](#)

[play_game](#) paper

[prize_amount](#)

problem3.sol

```

1 pragma solidity ^0.5.11;
2 contract rock_paper_scissors {
3     string public player1_choice;
4     string public player2_choice;
5     address payable public player1;
6     address payable public player2;
7     mapping (string => mapping(string => int)) winner_in_game;
8
9     constructor() public payable{
10         winner_in_game["rock"]["rock"] = 0;
11         winner_in_game["rock"]["paper"] = 2;
12         winner_in_game["rock"]["scissors"] = 1;
13         winner_in_game["paper"]["rock"] = 1;
14         winner_in_game["paper"]["paper"] = 0;
15         winner_in_game["paper"]["scissors"] = 2;
16         winner_in_game["scissors"]["rock"] = 2;
17         winner_in_game["scissors"]["paper"] = 1;
18         winner_in_game["scissors"]["scissors"] = 0;
19     }
20
21     modifier verify_enough_cash(uint amount){
22         if (msg.value < amount)
23             revert();
24         _;
25     }
26
27     modifier not_already_registered() {
28         if (player1 == msg.sender || player2 == msg.sender){
29             revert();
30         } else {
31             _;
32         }
33     }
34 }
35
36 function play_game(string memory _player1, string memory _player2) public not_already_registered() {
37     player1 = payable(msg.sender);
38     player2 = payable(msg.sender);
39     player1_choice = _player1;
40     player2_choice = _player2;
41     winner_in_game[player1_choice][player2_choice]++;
42 }

```

[address payable player1](#) 6 reference(s)

transact to rock_paper_scissors.play_game pending ...

[vm] from: 0x787...cabab to: rock_paper_scissors.play_game(string) 0xd91...39138 value: 0 wei data: 0x0c1...00000 logs: 0
hash: 0xa9c...60fa4 [Debug](#)

transact to rock_paper_scissors.get_balance pending ...

Part 4) Register 2 players with 5 ethers and give their choices.

The screenshot displays the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel shows the 'ROCK_PAPER_SCISSORS' contract deployed at address 0xD91...39138. The 'Deployed Contracts' section lists several functions: 'get_balance', 'play_game' (with a dropdown menu set to 'paper'), 'prize_amount', and 'register_player'. Below these, the 'player1' and 'player2' addresses are shown, along with their respective choices: 'rock' for player1 and 'paper' for player2. The 'Low level interactions' section shows a 'CALLDATA' field and a 'Transact' button.

The main editor displays the Solidity code for the 'rock_paper_scissors' contract. The code includes a pragma statement, contract definition, public variables for player choices, payable addresses for players, a mapping for game results, a constructor to initialize the game state, and two modifiers: 'verify_enough_cash' and 'not_already_registered'.

```
1 pragma solidity ^0.5.11;
2 contract rock_paper_scissors {
3     string public player1_choice;
4     string public player2_choice;
5     address payable public player1;
6     address payable public player2;
7     mapping (string => mapping(string => int)) winner_in_game;
8
9     constructor() public payable{
10         winner_in_game["rock"]["rock"] = 0;
11         winner_in_game["rock"]["paper"] = 2;
12         winner_in_game["rock"]["scissors"] = 1;
13         winner_in_game["paper"]["rock"] = 1;
14         winner_in_game["paper"]["paper"] = 0;
15         winner_in_game["paper"]["scissors"] = 2;
16         winner_in_game["scissors"]["rock"] = 2;
17         winner_in_game["scissors"]["paper"] = 1;
18         winner_in_game["scissors"]["scissors"] = 0;
19     }
20
21     modifier verify_enough_cash(uint amount){
22         if (msg.value<amount)
23             revert();
24         else
25             _;
26     }
27
28     modifier not_already_registered() {
29         if(player1==msg.sender || player2==msg.sender){
30             revert();
31         } else {
32             _;
33         }
34     }
35 }
```

The bottom panel shows a transaction log with the following entry:

```
transact to rock_paper_scissors.play_game pending ...
[vm] from: 0x787...cabab to: rock_paper_scissors.play_game(string) 0xd91...39138 value: 0 wei data: 0x0c1...00000 logs: 0
hash: 0xa9c...60fa4
```

Part 5) Check the balance at the end of the game. As we can see the second player won so he has 104.99 ether and player 1 has 94.99 ether.

JavaScript VM

0x5B3...eddc4 (99.9999999999998983227 ether)

0xab8...35cb2 (94.9999999999998983227 ether)

0x4B2...C02db (94.999999999999908652 ether)

✓ 0x787...cabaB (104.999999999999871778 ether)

0x617...5E7f2 (100 ether)

0x17F...8c372 (100 ether)

0x5c6...21678 (100 ether)

0x03C...D1Ff7 (100 ether)

0x1aE...E454C (100 ether)

0x0A0...C70DC (100 ether)

0xCA3...a733c (100 ether)

0x147...C160C (100 ether)

0x4B0...4D2dB (100 ether)

0x583...40225 (100 ether)

0xdD8...92148 (100 ether)

Deployed Contracts



▼ ROCK_PAPER_SCISSORS AT 0X1BB...87B0B (MEMORY)

play_game

string choice



prize_amount

register_player

player1

player1_choice

player2

player2_choice

Low level interactions



Part 4) Register 2 players