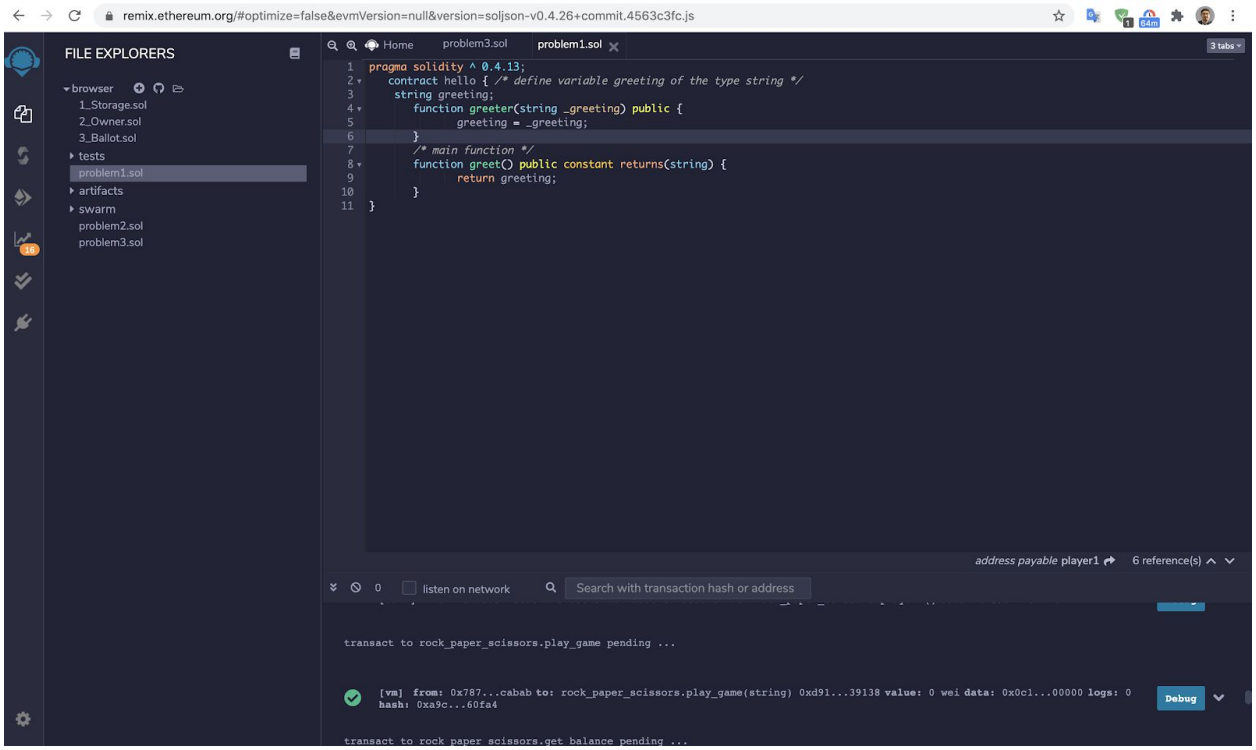
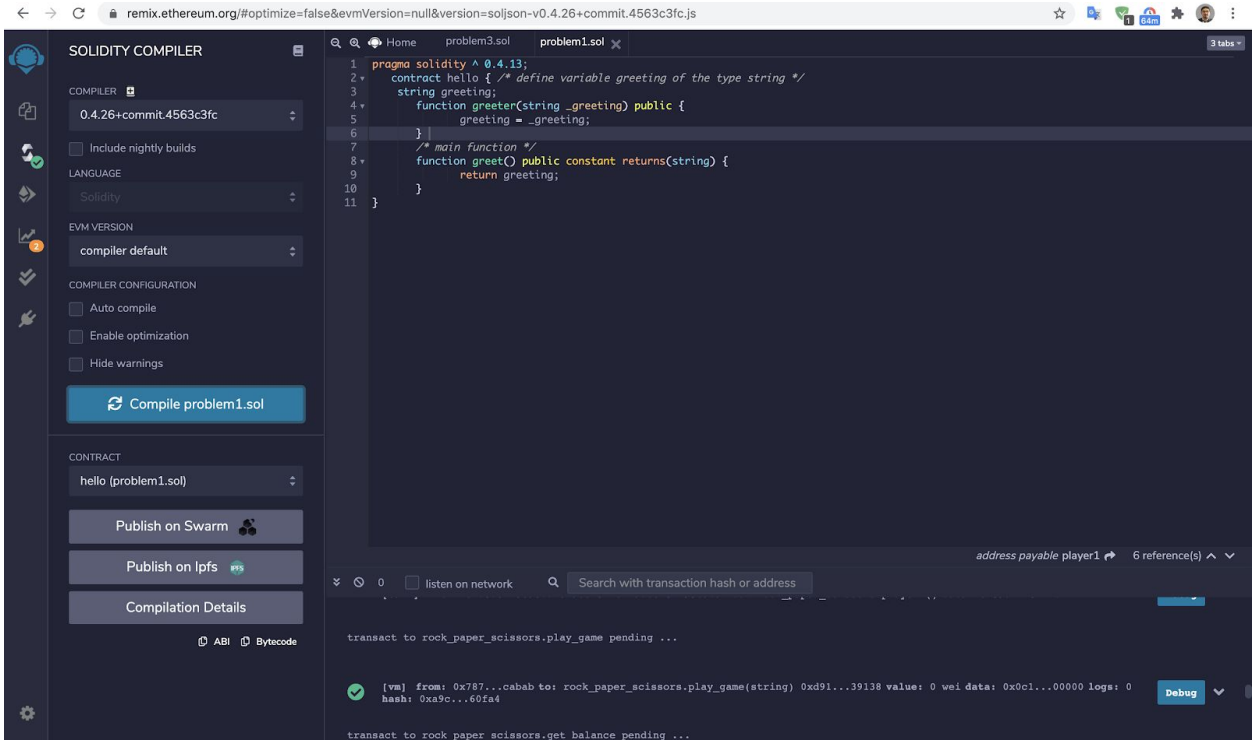
Problem 1

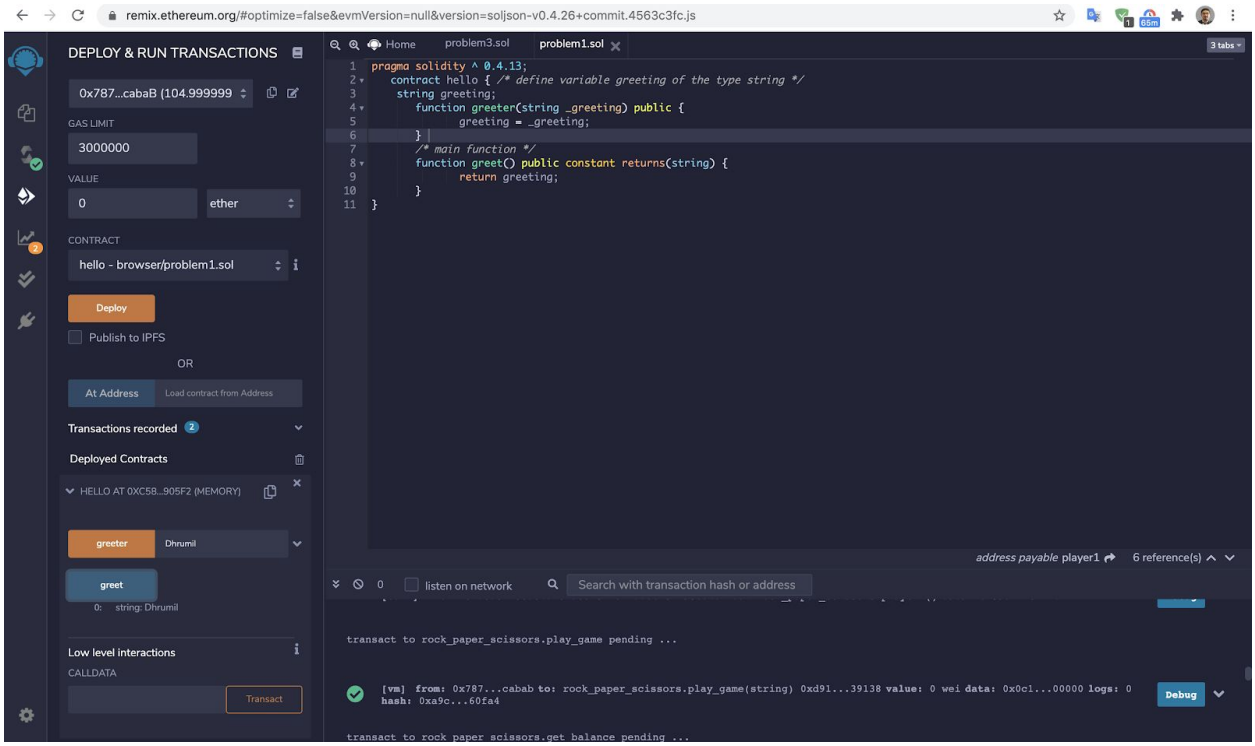
Part -1) Code



Part 2) Compile

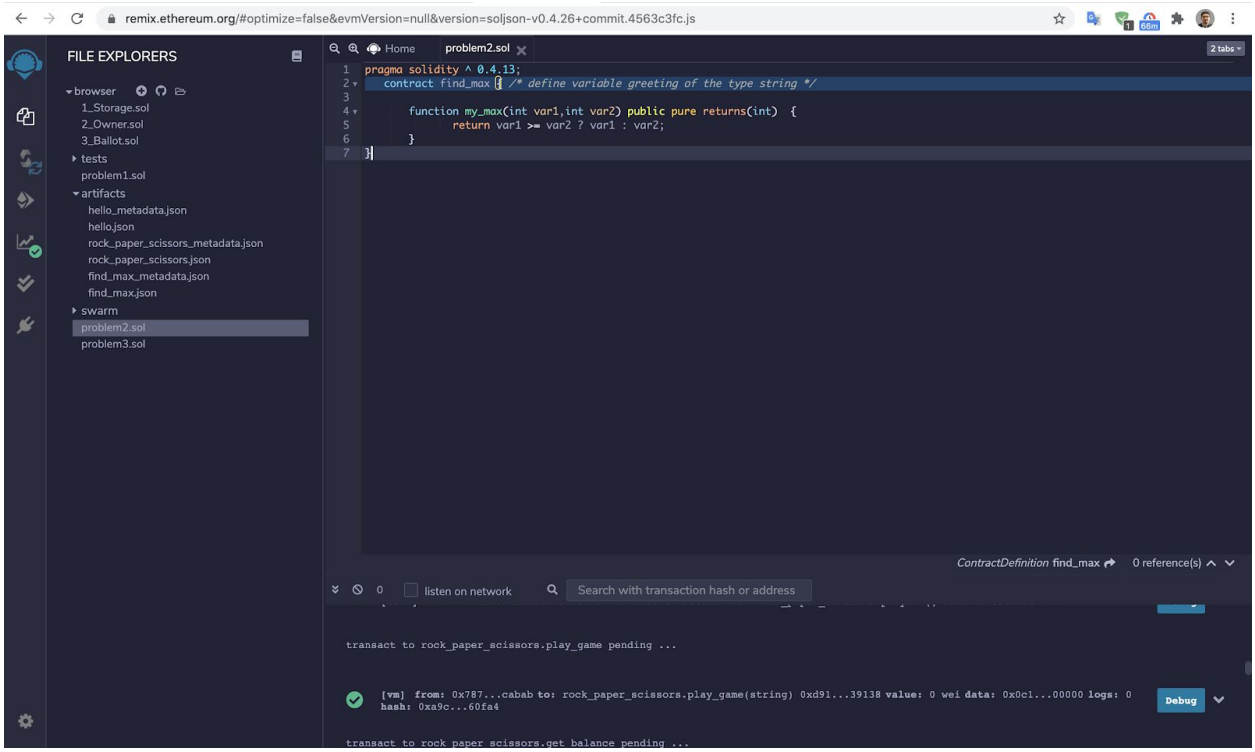


Part 3) Deploy and output

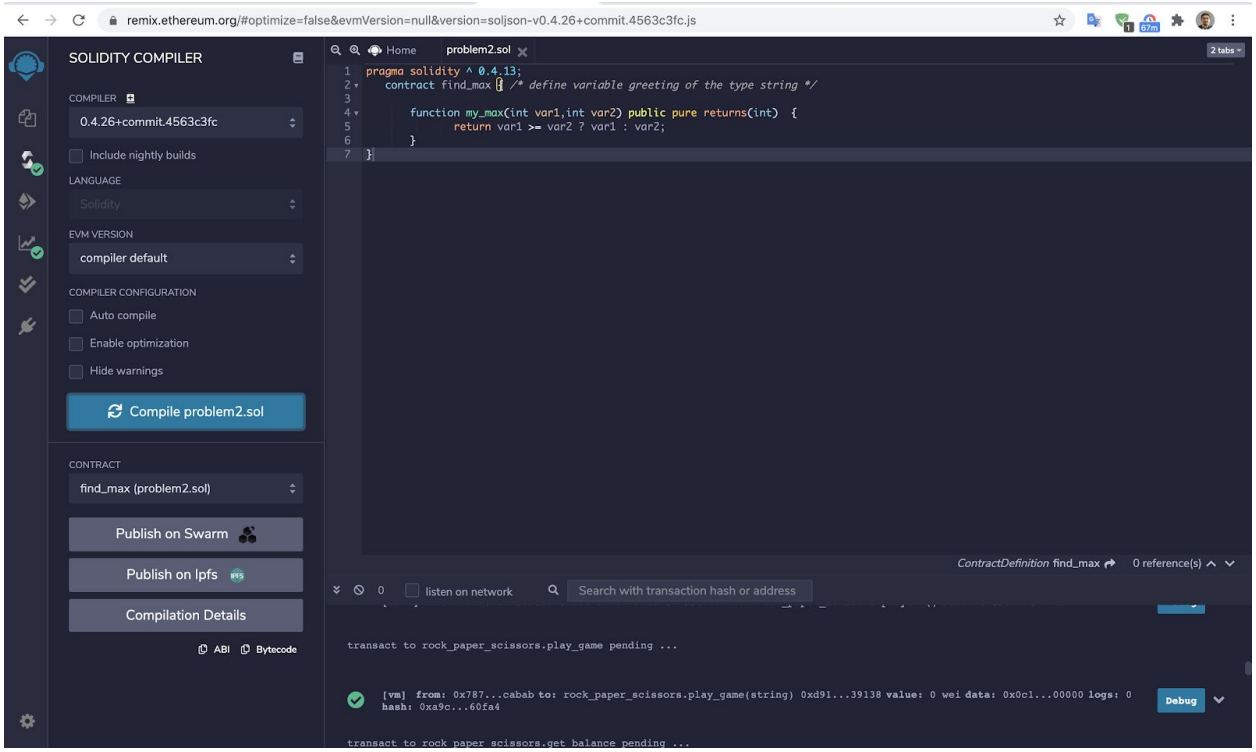


Problem 2):

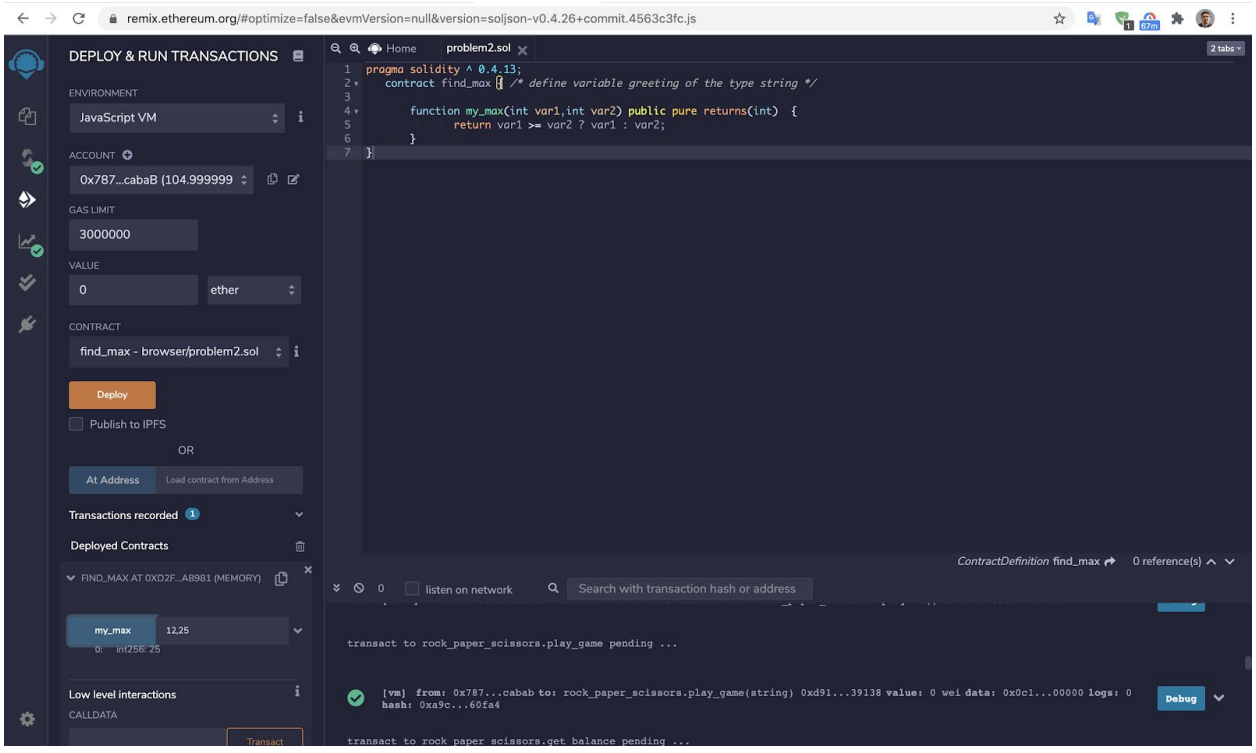
Part 1) Code



Part 2) Compile



Part 3) Deploy and test



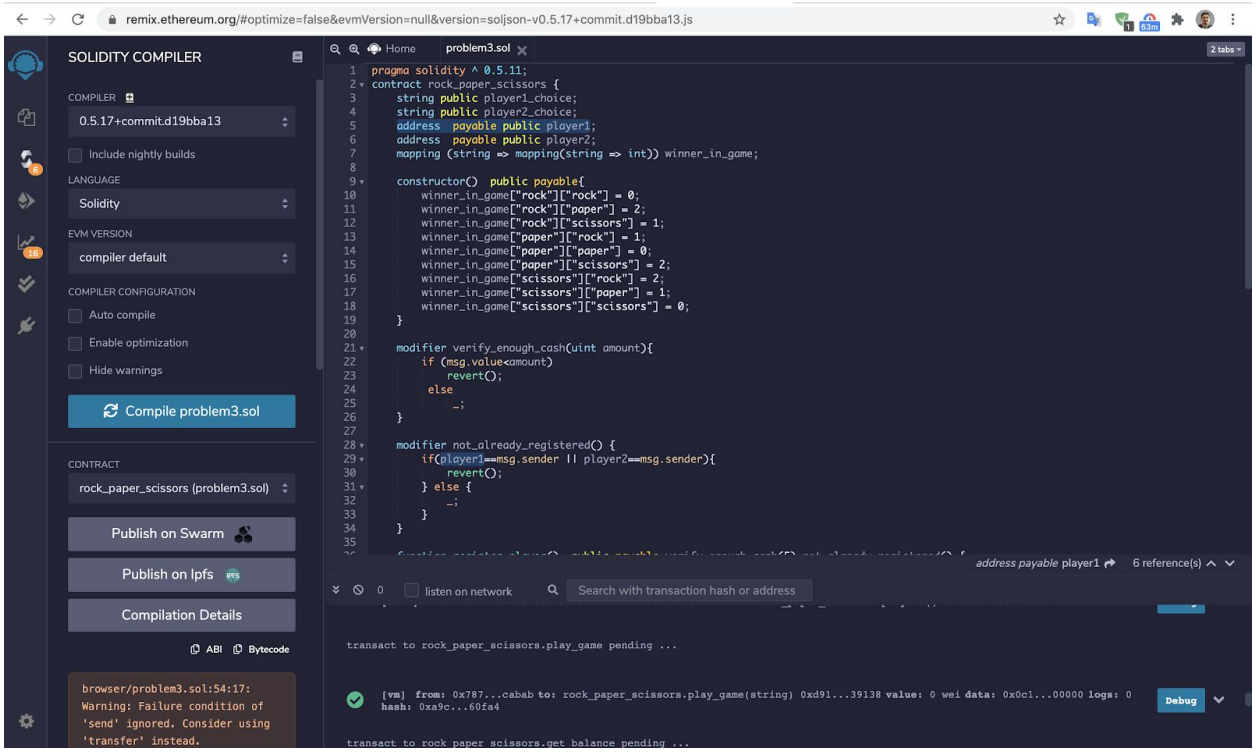
Problem 3 :

Part 1) Solidity Program:

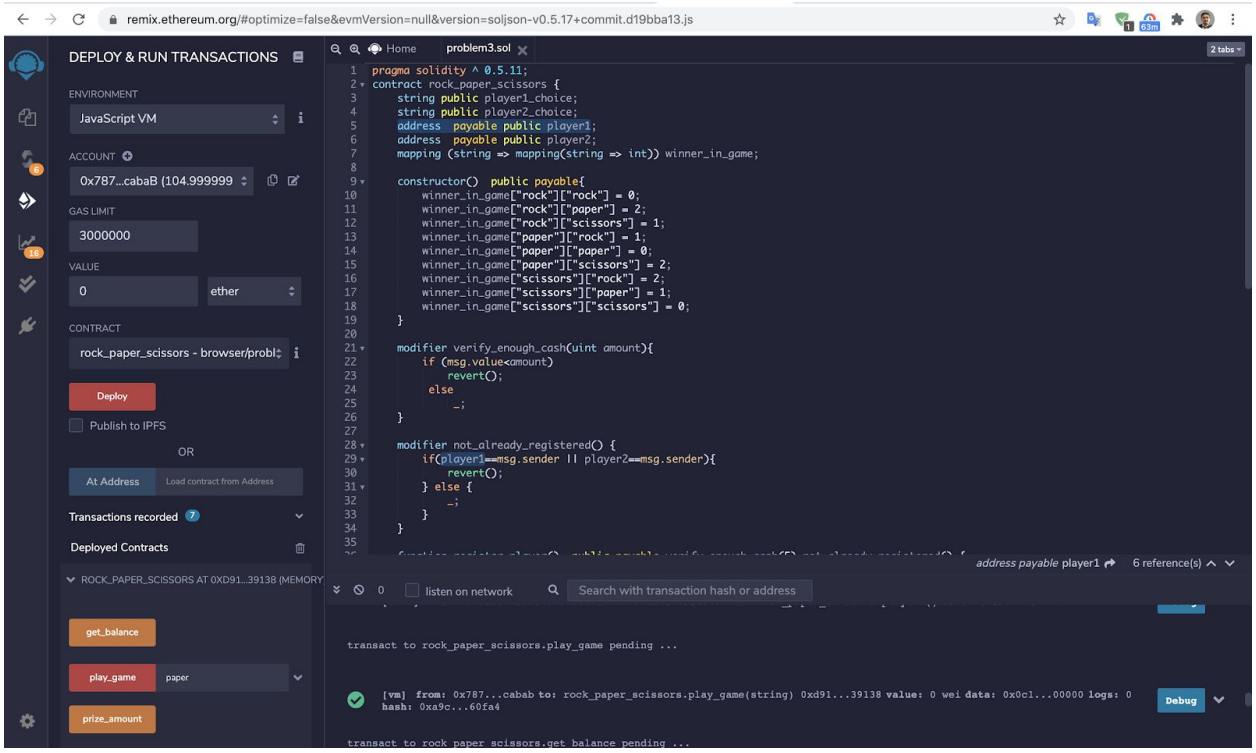
```
pragma solidity ^ 0.5.11;
contract rock_paper_scissors {
    string public player1_choice;
    string public player2_choice;
    address  payable public player1;
    address  payable public player2;
    mapping (string => mapping(string => int)) winner_in_game;

    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;
    }


}
```

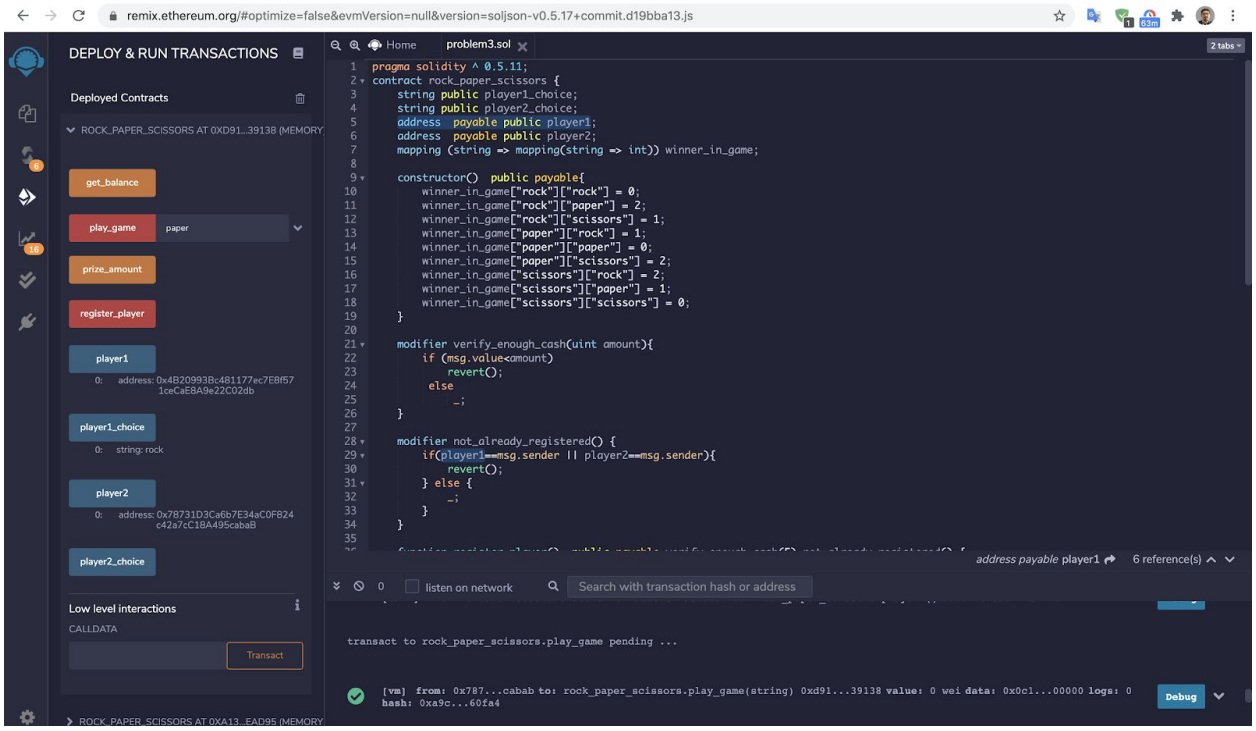
Part 2) Compile the program:



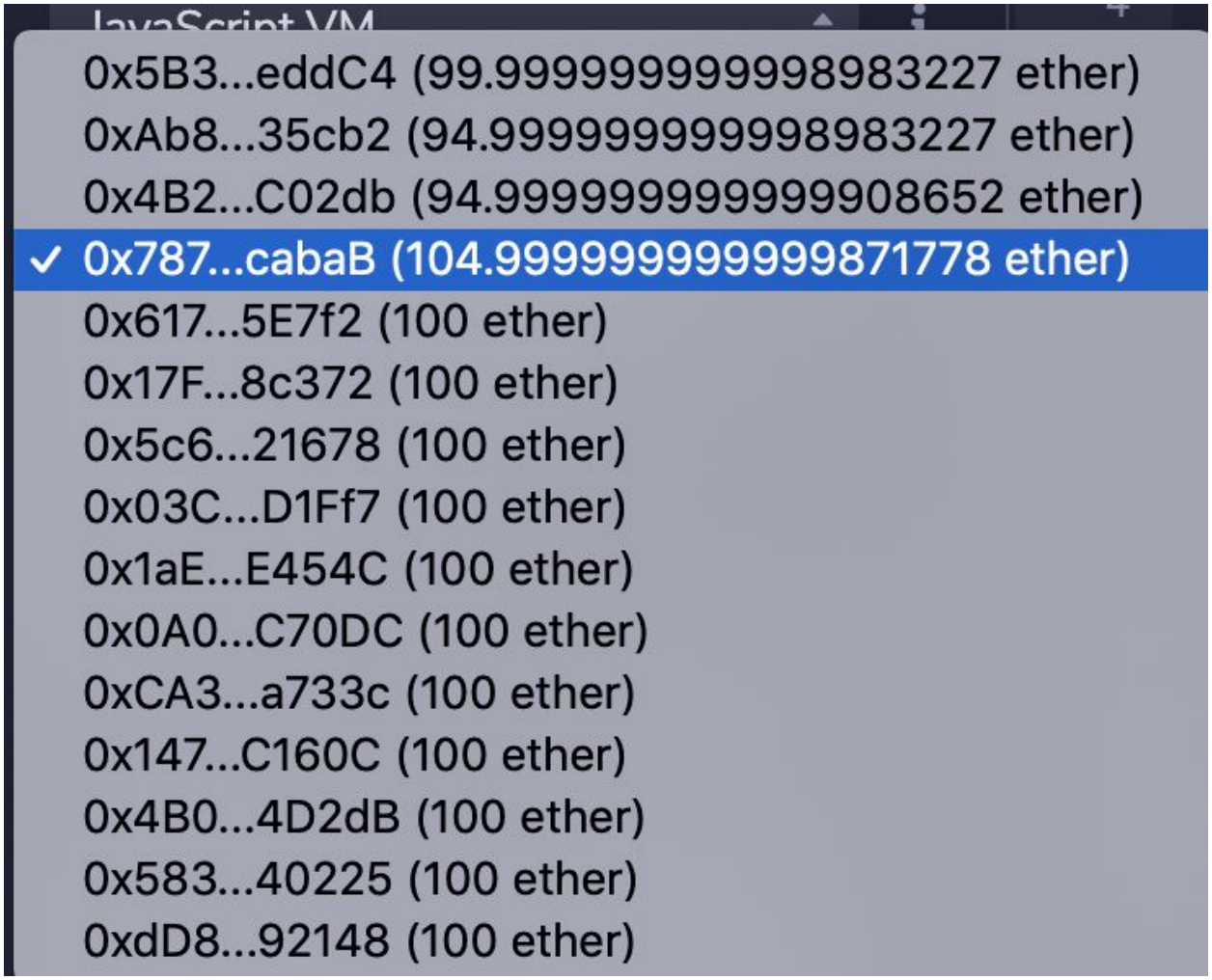
Part 3) Deploy the program :

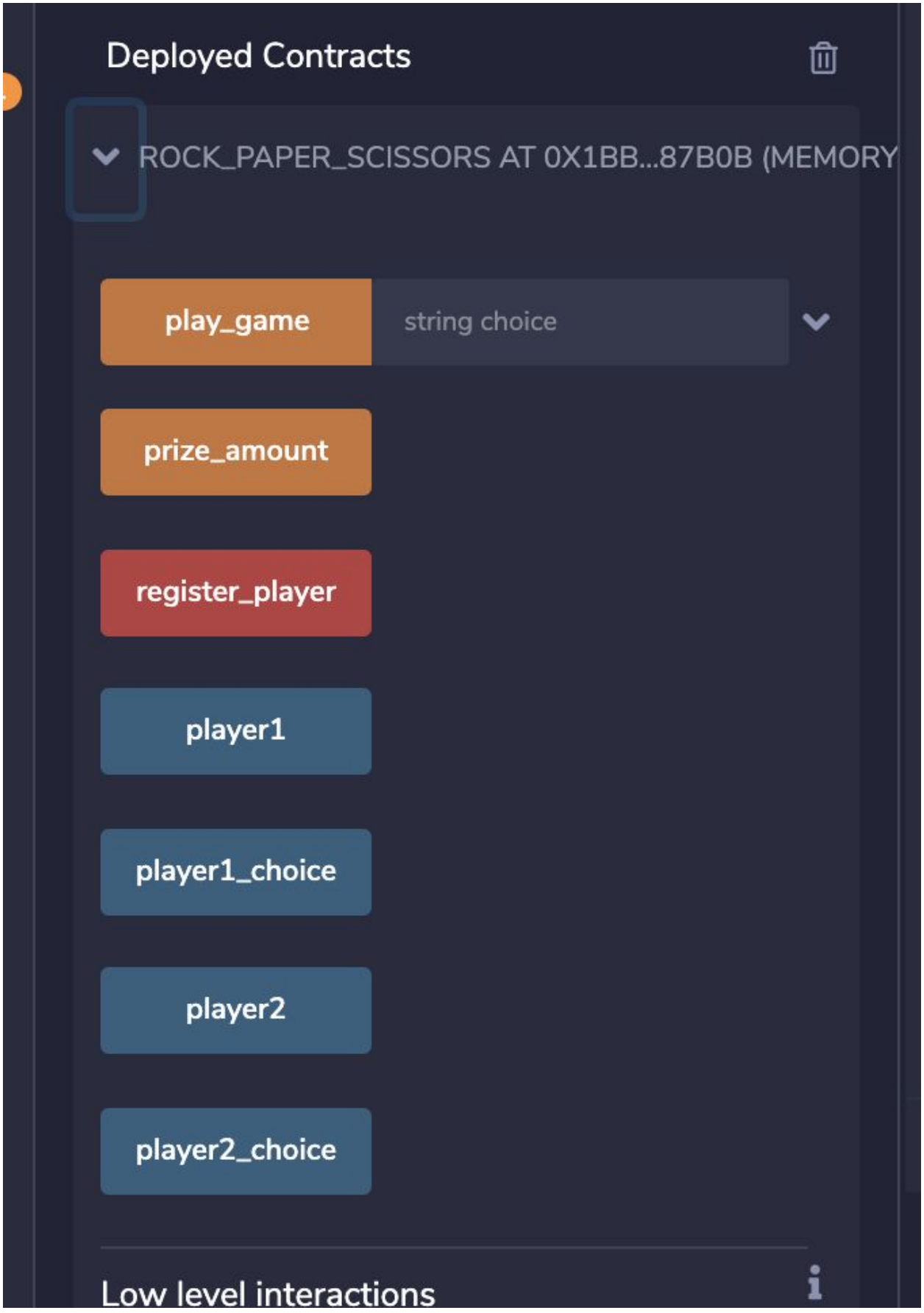


Part 4) Register 2 players with 5 ethers and give their choices.



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


JavaScript VM
0x5B3...eddC4 (99.99999999998983227 ether)
0xAb8...35cb2 (94.99999999998983227 ether)
0x4B2...C02db (94.99999999999908652 ether)
0x787...cabaB (104.99999999999871778 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