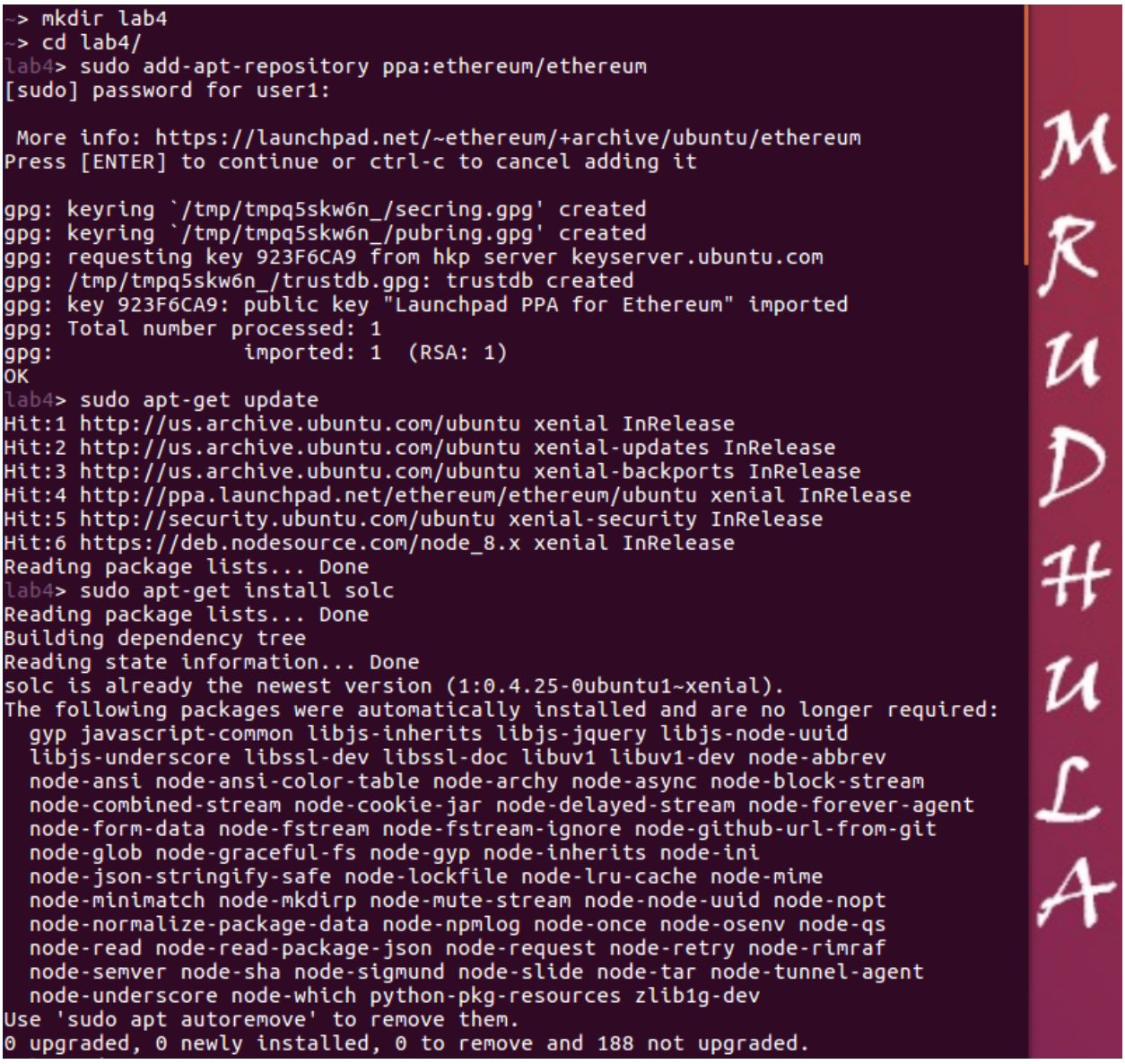
SMART CONTRACT PROGRAMMING

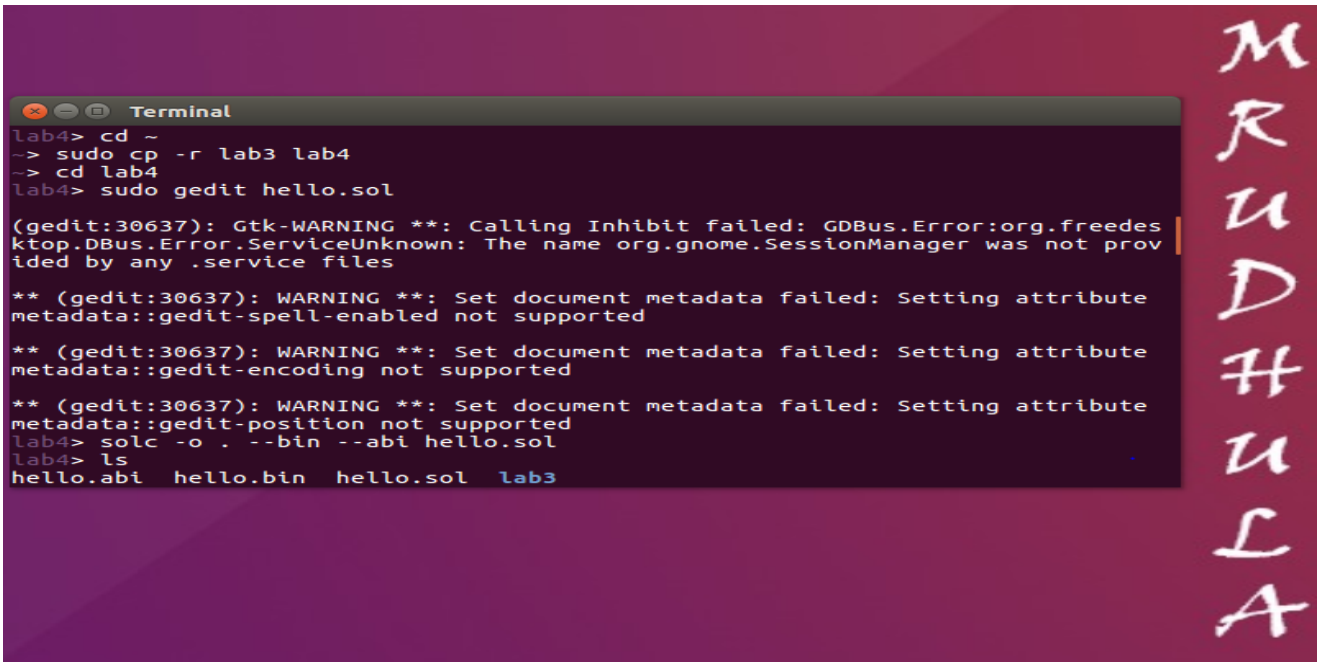
INITIAL SETUP



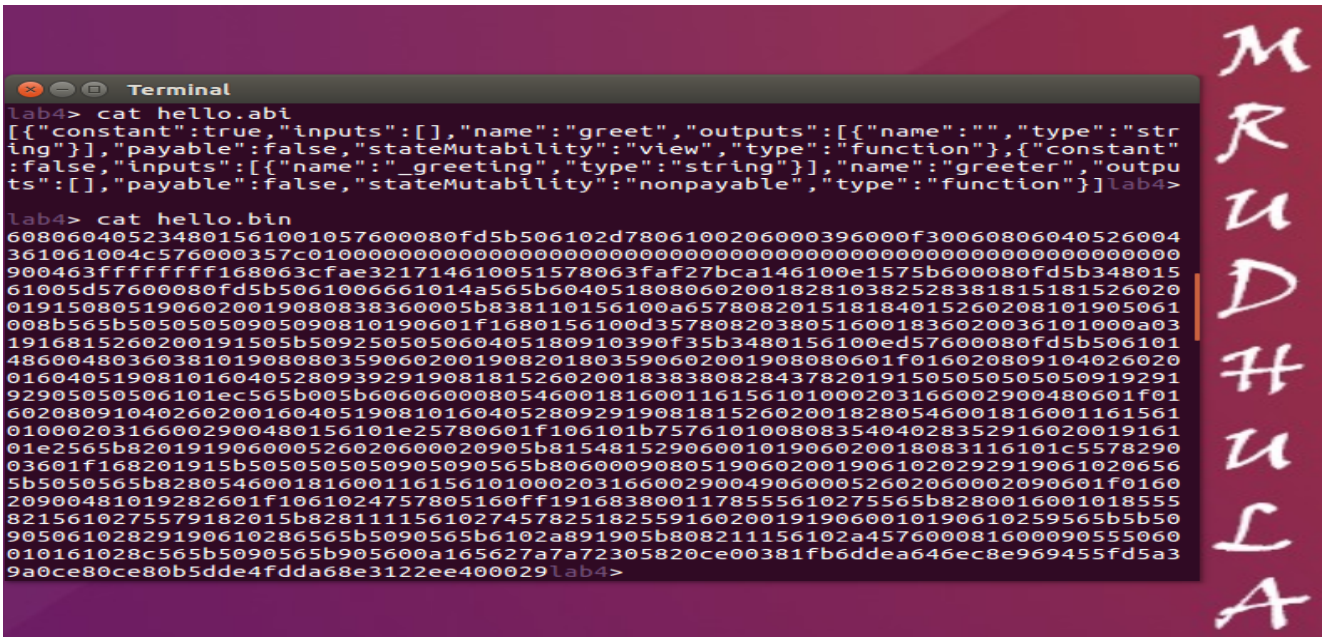
We install Solc and set up the Ethereum client for this lab. We use the following commands for Solc installation:

sudo add-apt-repository ppa:Ethereum/Ethereum
sudo apt-get update
sudo apt-get install solc

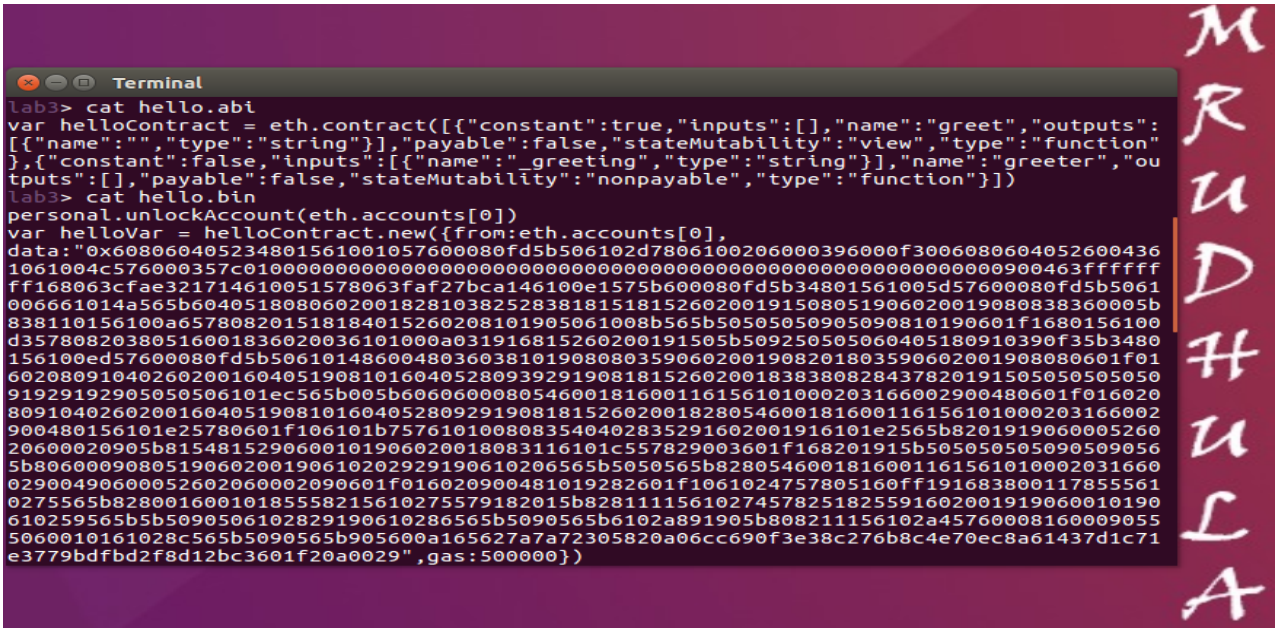
EXERCISE 1a: Hello-world Contract with Remix



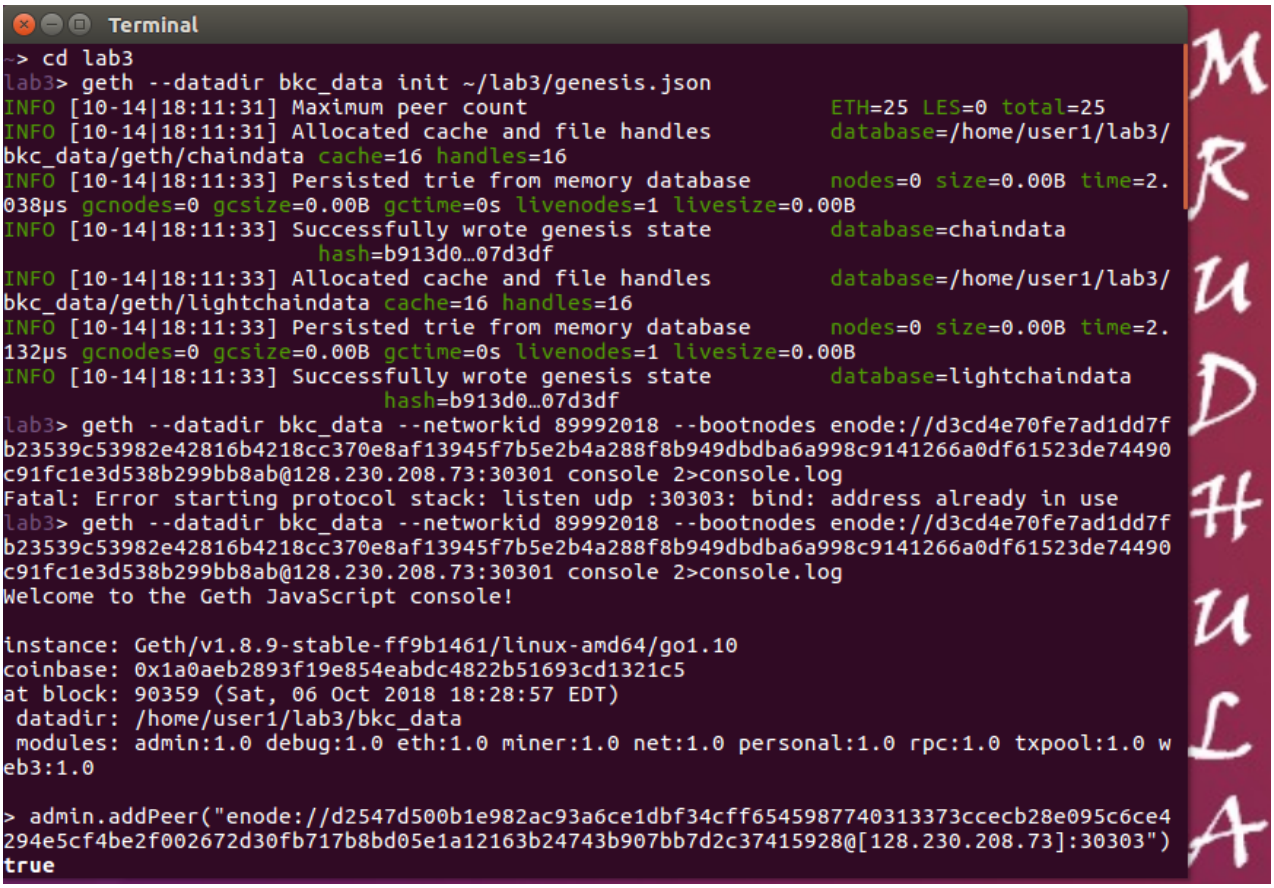
We create a smart contract file and copy the text of hello world program provided to us and save it with an extension of '.sol', then we compile the code. Through the compilation we will be generating two files one 'hello.abi" (it is an interface through which the compiled codes interact with the execution platform) and another file is 'hello.bin' this file stores the binary code.



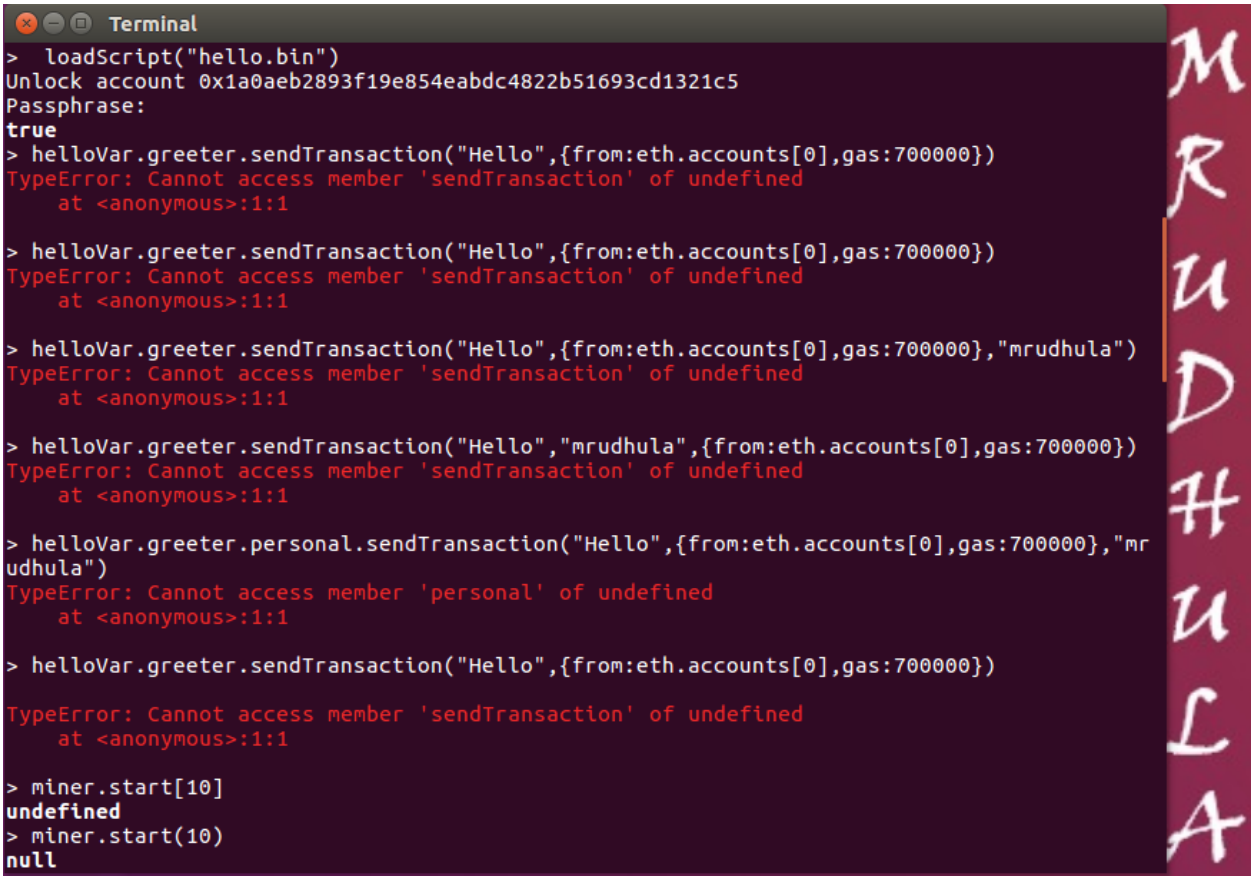
The above screenshot displays the original contents of the two files generated after compilation.



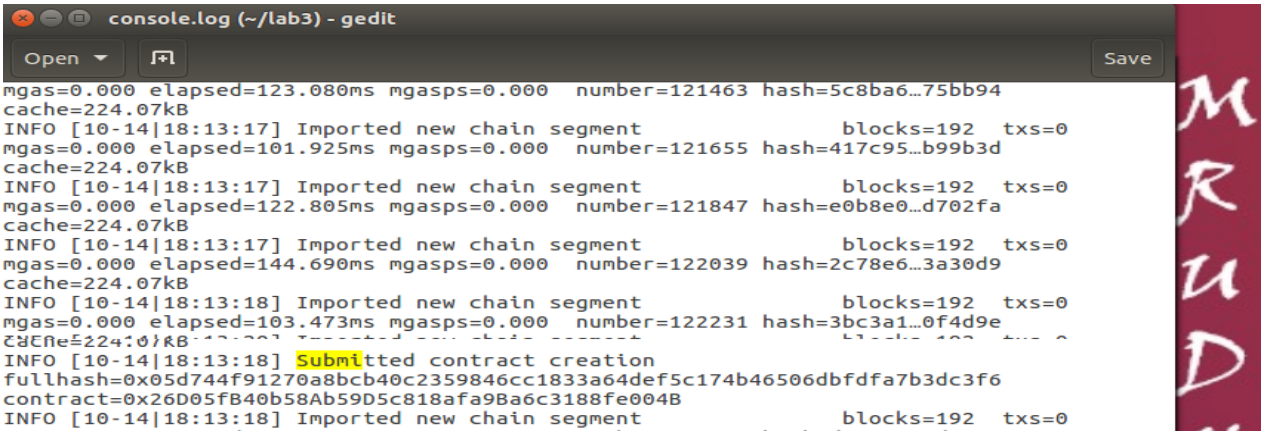
To deploy the compiled files, we modify the contents of both the generated files as shown in the above screenshot.

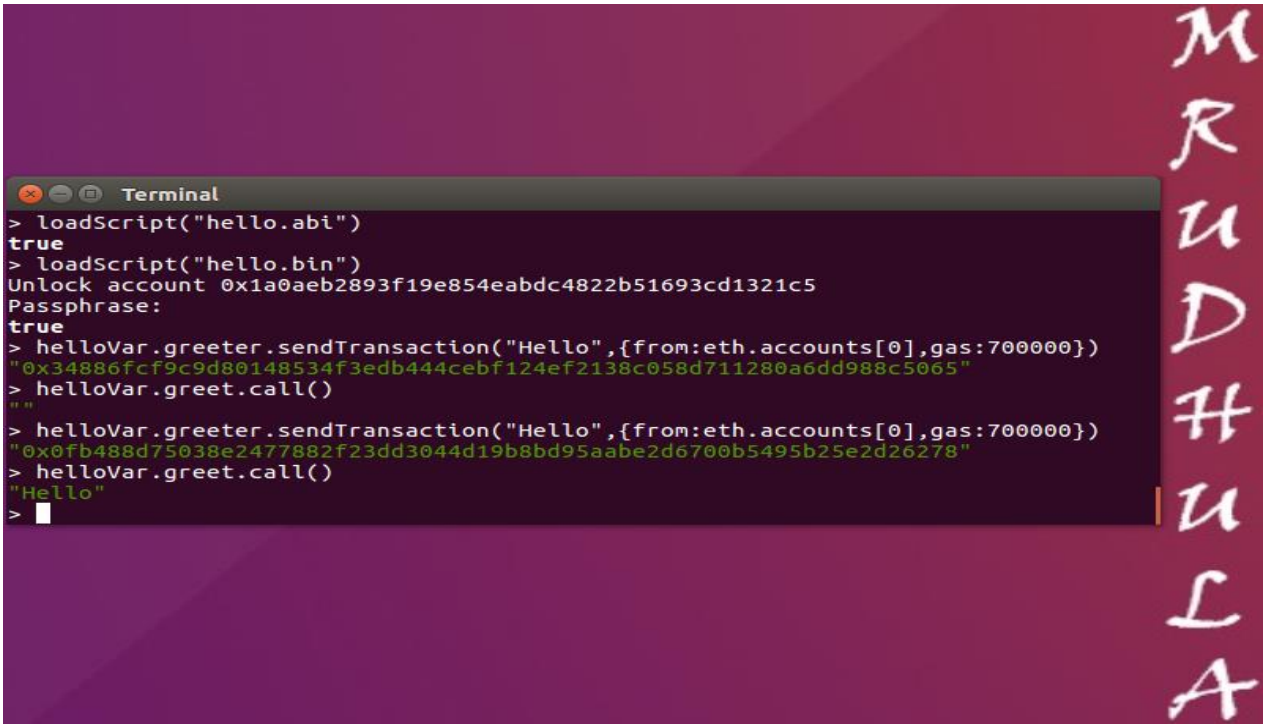


We deploy the smart contract by connecting to the Ethereum network as shown above.



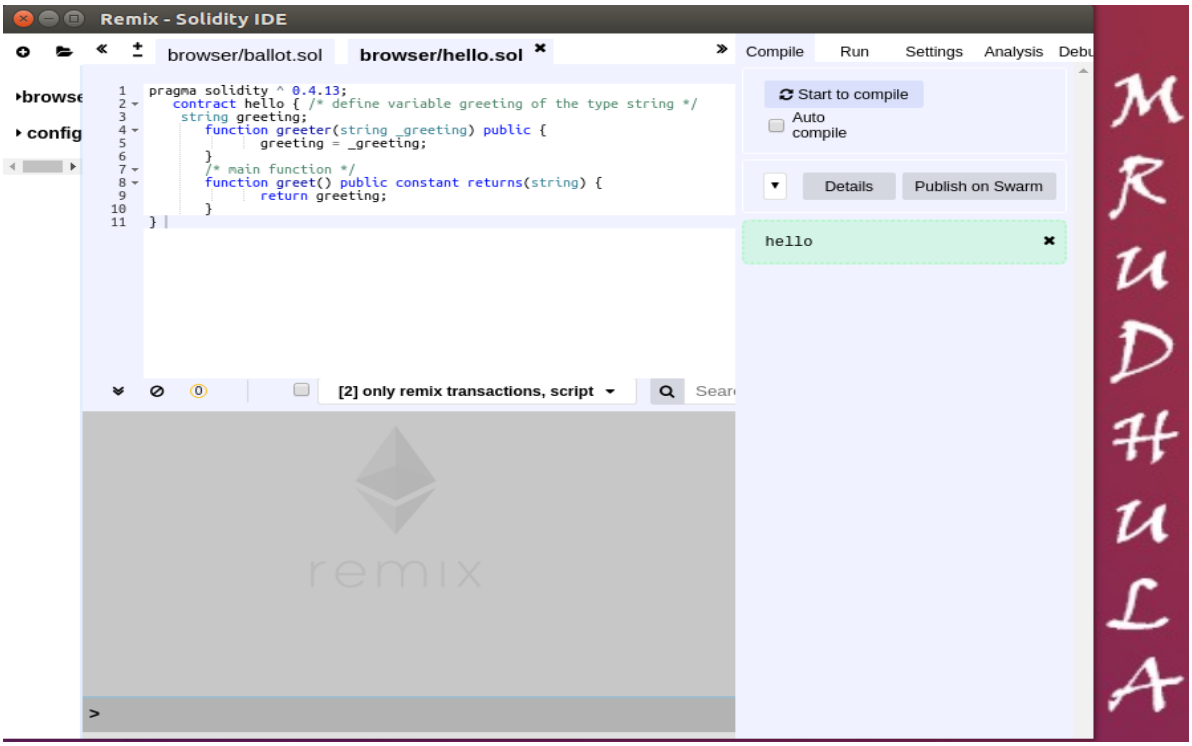
Then we load and run the script to deploy the smart contract. When we load and run 'hello.abi' it gives true and when we load and run 'hello.bin' it unlocks our account and asks for our passphrase. Then when we check the console log file we find a contract address and transaction hash value as shown below. Now we do send transaction giving the contract name and the initial function 'greeter' name through with we pass the arguments. As I tried multiple times I notice that I do not get any results. Then we start mining to make the process faster and start the send transaction again, we keep trying till we achieve a transaction hash value. If suppose we do not achieve in initially try (In my case) I reloaded and re-ran the script to deploy the smart contract after I started mining and then when I tried the send transaction it was successful, and I got a transaction hash value back. As shown below the screenshot of console log file .



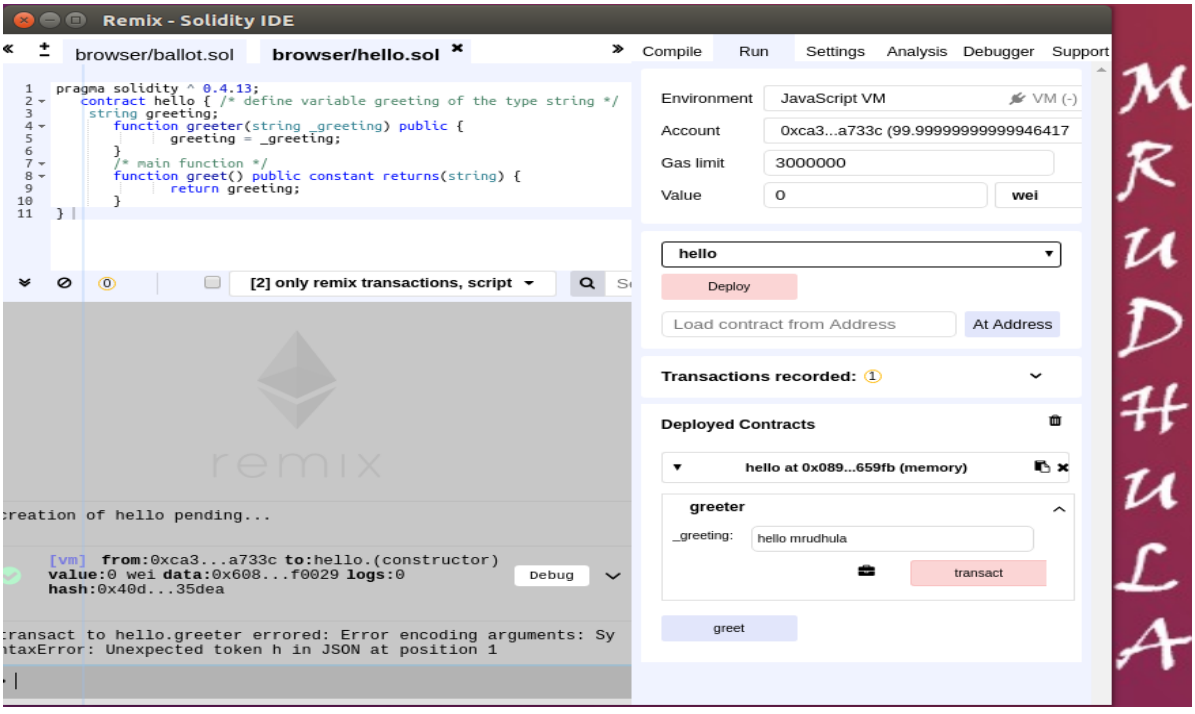


We execute the greet function in which we get the hello displayed. We check our console log file ones again to see the new contract address and the transaction hash value after executing the functions sequentially.

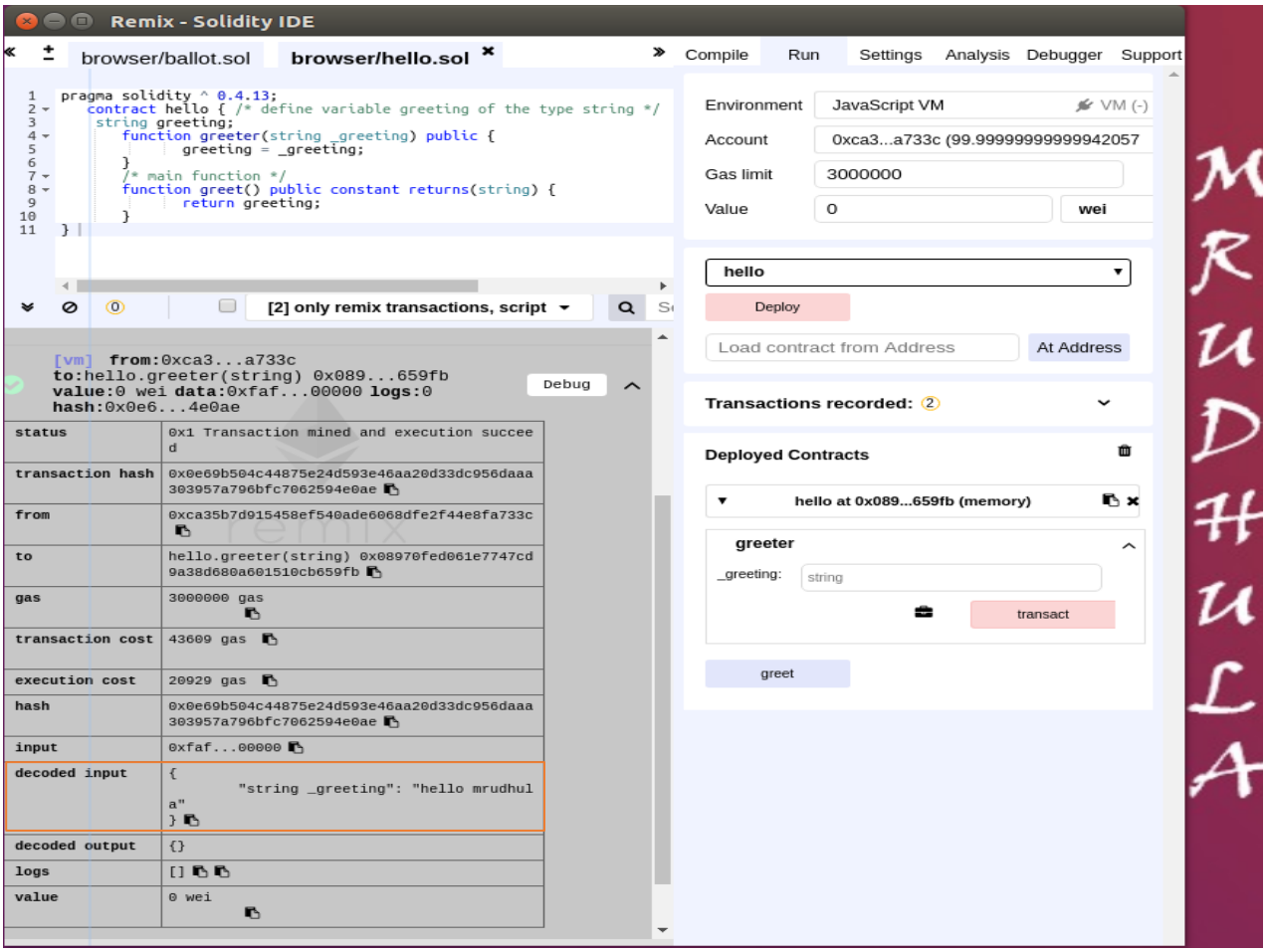
EXERCISE 1b: Hello-world Contract with Solc and On-Campus Ethereum



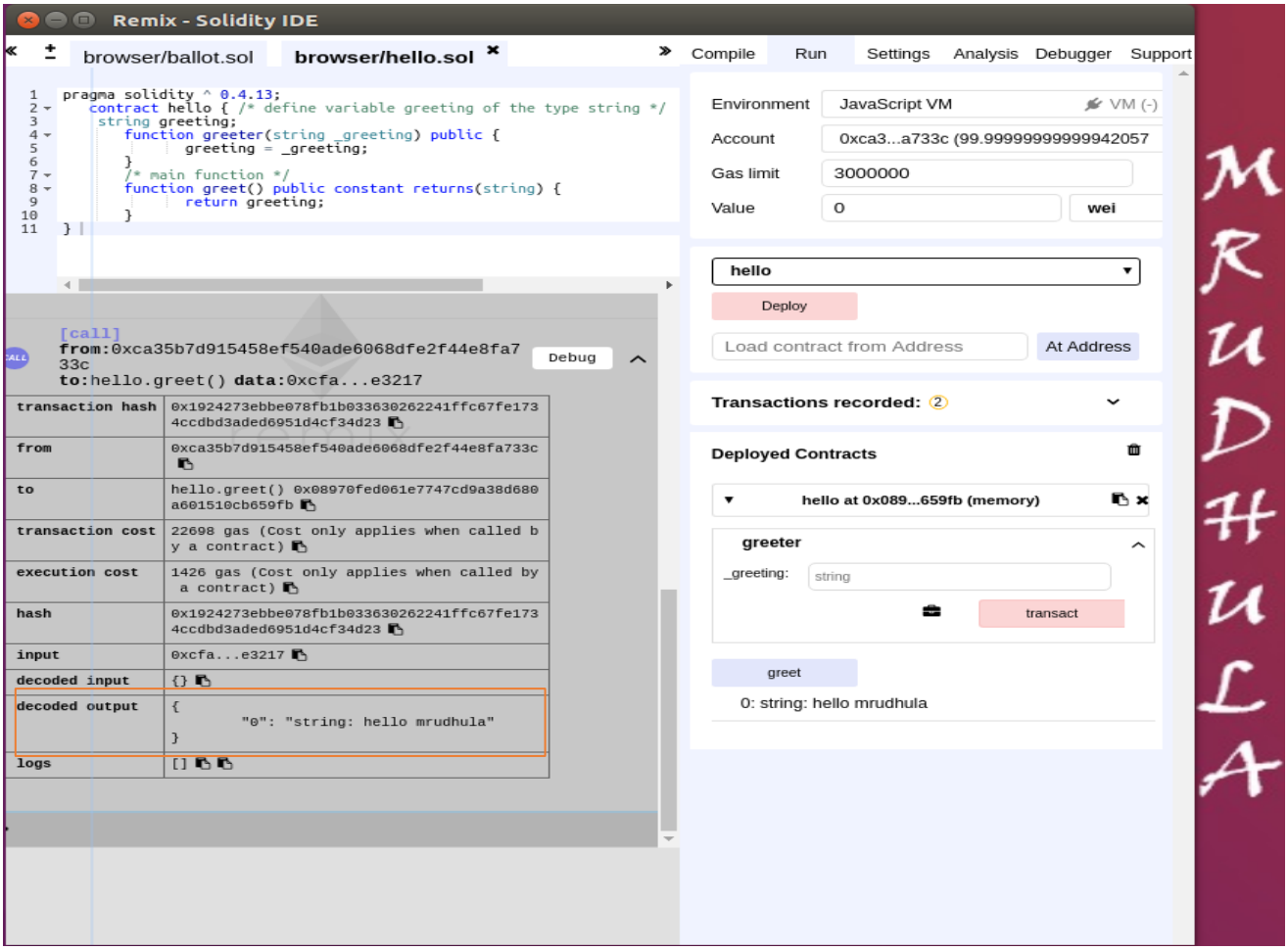
We run the given hello-world program on a command-line programming framework solc. We first compile the program, when program is successfully compiled it will return the program name.



Then we run the program after compiling it successfully. We get an option bar deploy and we click on that to deploy the contract. We give a value for the first function greeter. And click on transact.



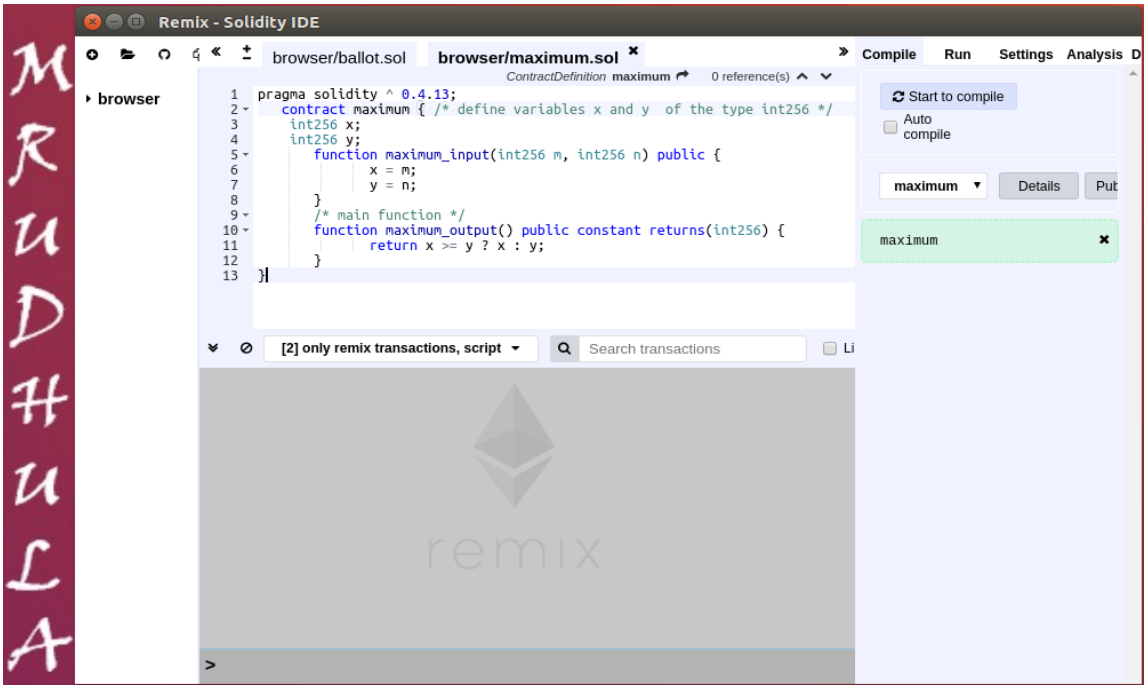
After we transact, we can see the transaction on the output screen and it shows the decoded input.



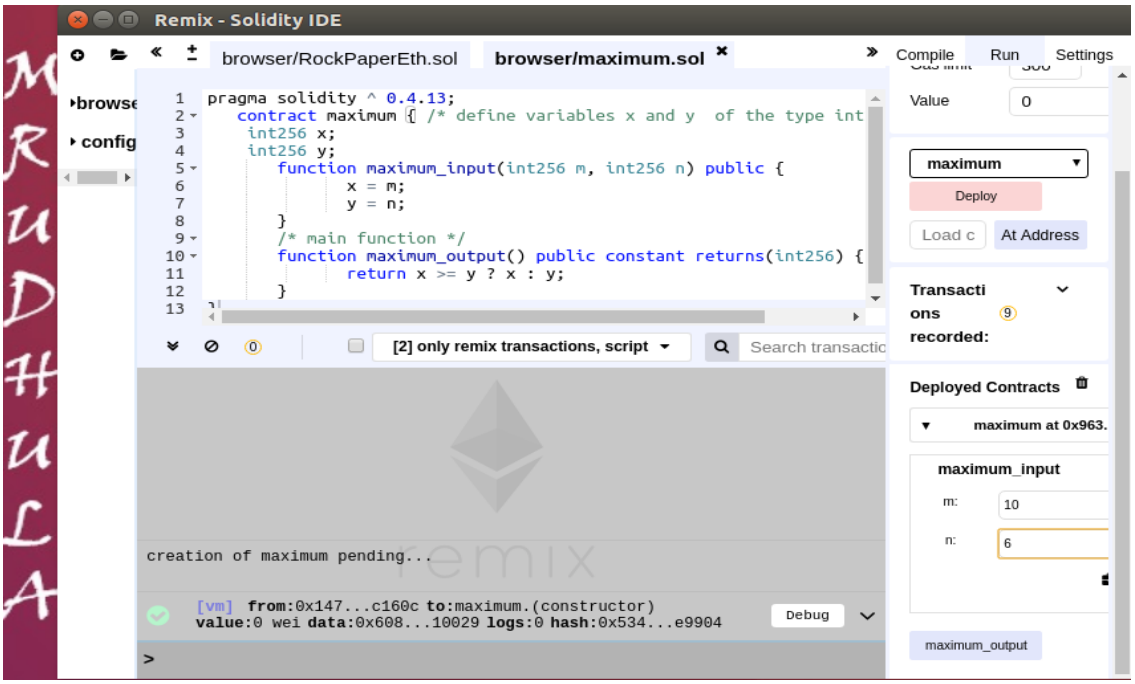
After that we click on the greet function option which will provide us the decoded output. We can check this on the output screen as shown above which will print the string that we provided to the greeter function.

EXERCISE 2: Find the Maximum

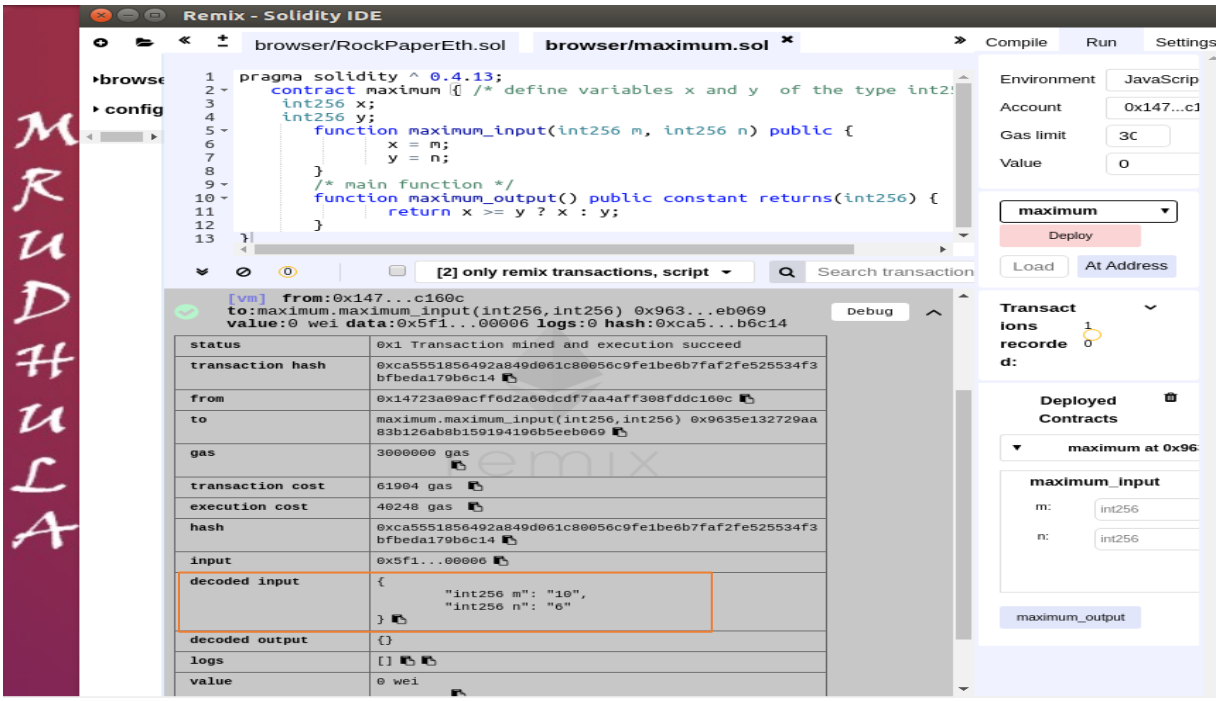
In Solc



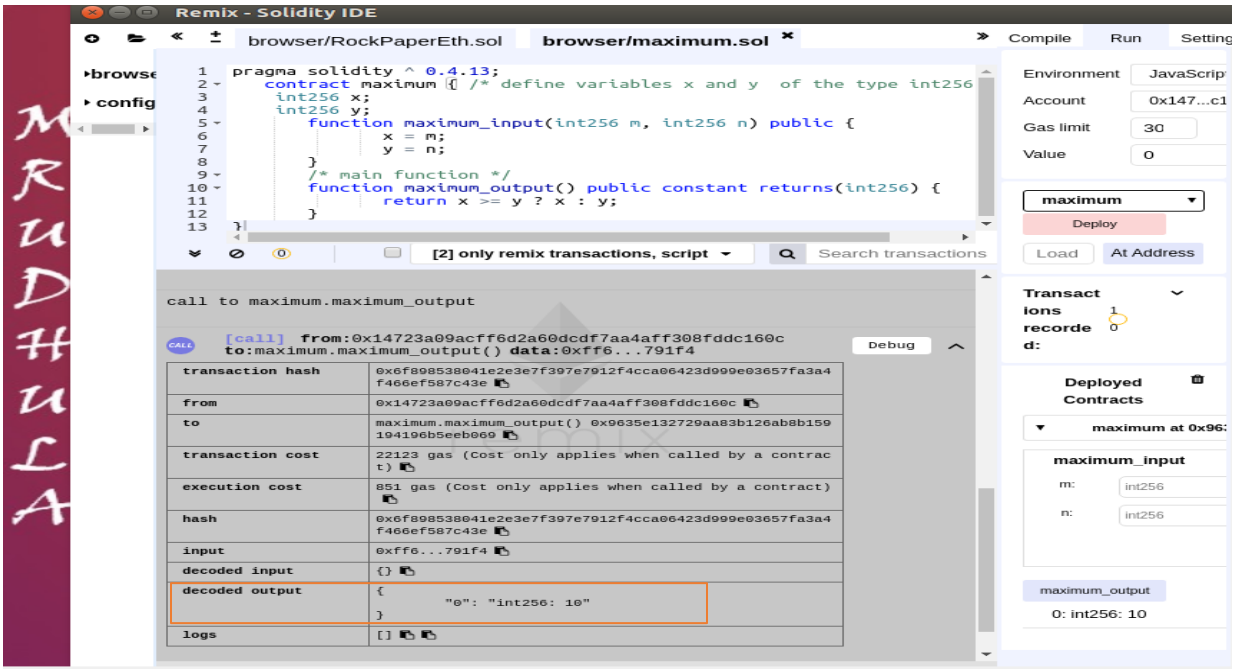
We write program for finding the maximum numbers and we take an input of x and y. The two functions here are maximum_input and maximum_output. In maximum_output function we give a condition to check both the values and print out the maximum value. We compile the program. We see the program is successfully compiled as it returns the program name in green.



We then deploy the contract we give details for the first function that is maximum_input which takes two values which are of integer type.



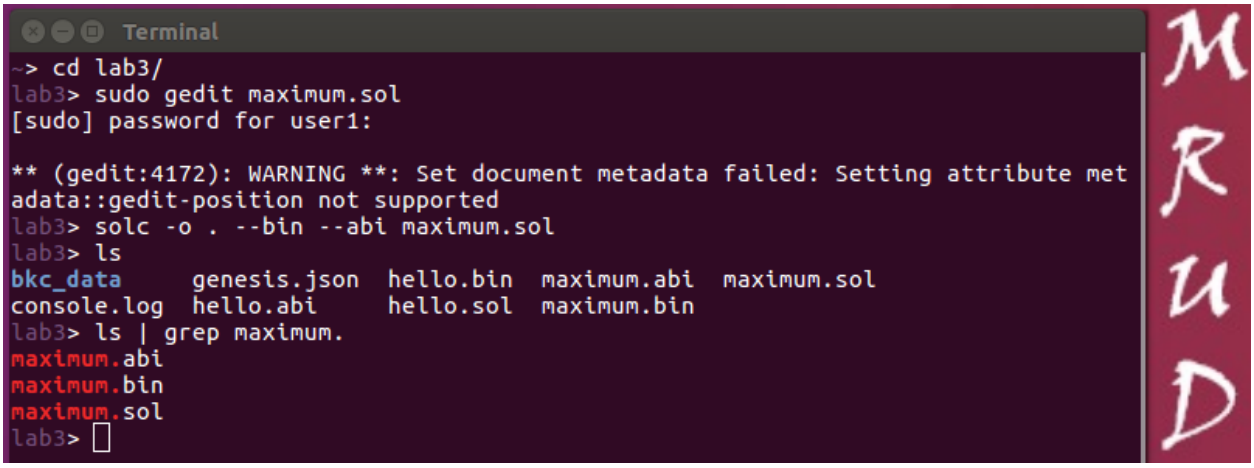
We transact the function and check the console log and we see in the decoded input are the two value that we had given to find the maximum value.

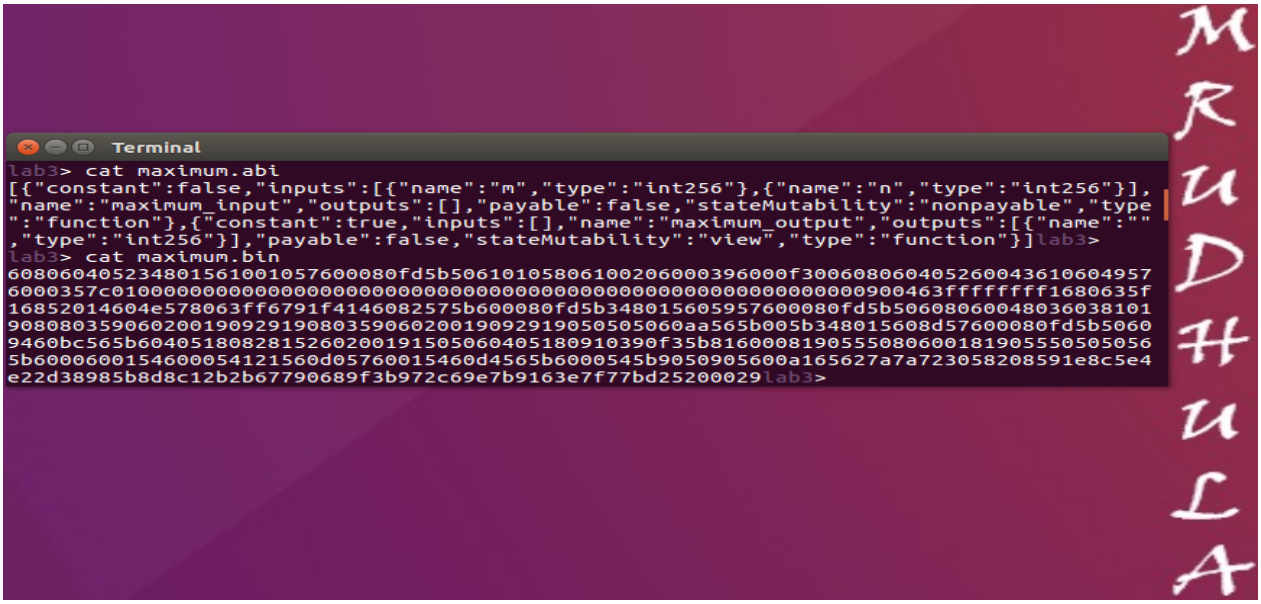


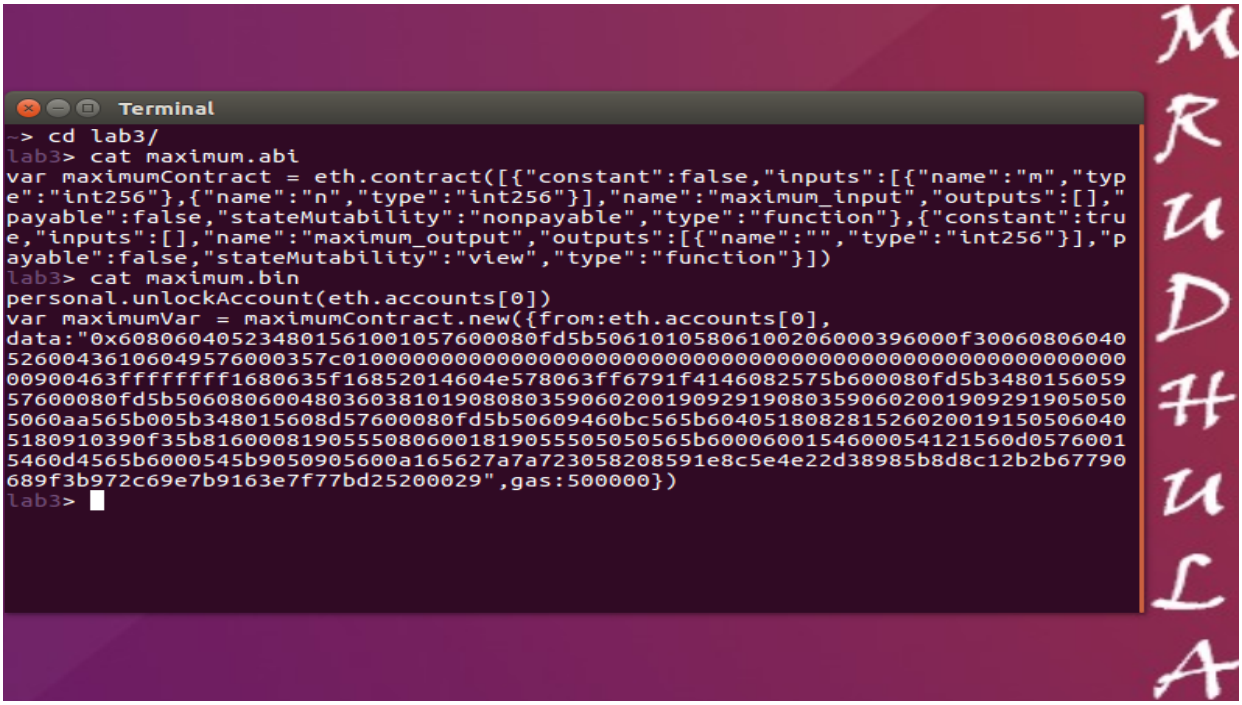
We now click on the next function that is the maximum_output function we see that we have obtained the maximum value out of the two integer numbers we had passed. We check the console log to see if our answer is recorded and visible at the decoded output as shown above.

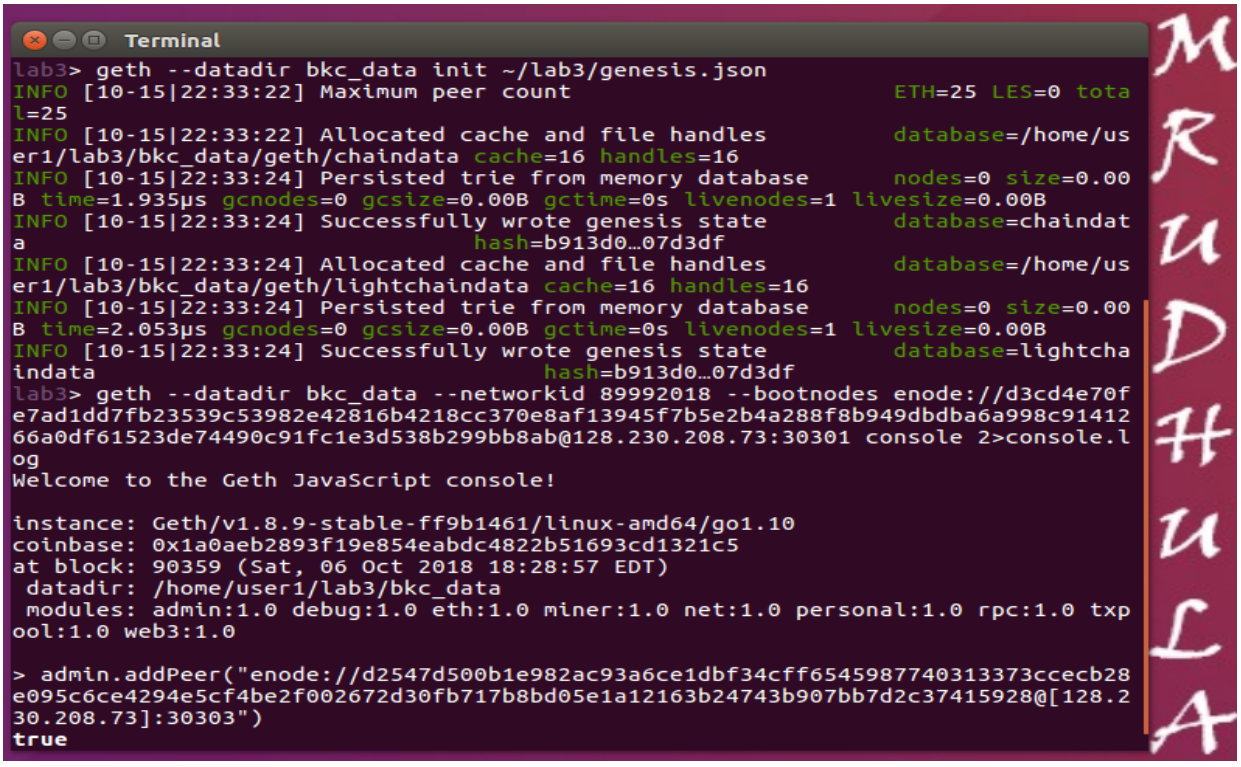
On Terminal

We write the same program that we did in solc. And repeat the steps done for exercise 1a.



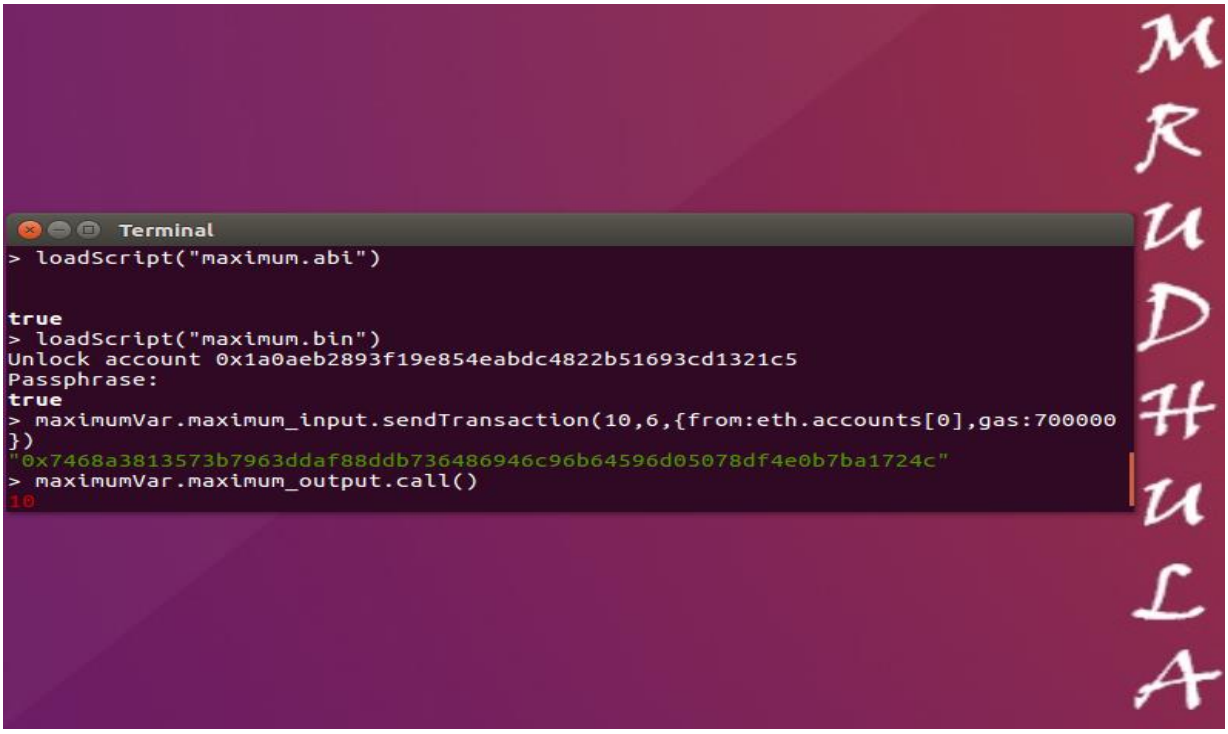



Terminal
~> cd lab3/
lab3> cat maximum.abi
var maximumContract = eth.contract([{"constant":false,"inputs":[{"name":"m","type":"int256"},{"name":"n","type":"int256"}],"name":"maximum_input","outputs":[],"payable":false,"stateMutability":"nonpayable","type":"function"},{"constant":true,"inputs":[],"name":"maximum_output","outputs":[{"name":"","type":"int256"}],"payable":false,"stateMutability":"view","type":"function"}])
lab3> cat maximum.bin
personal.unlockAccount(eth.accounts[0])
var maximumVar = maximumContract.new({from:eth.accounts[0],
data:"0x6080604052348015610010576000080fd5b506101058061002060000396000f30060806040526004361060495760003557c0100000000000000000000000000000000000000000000000000000000900463ffffffff1680635f16852014604e578063ff6791f4146082575b600080fd5b3480156059576000080fd5b506080600480360381019080803590602001909291908035906020019092919050505060aa565b005b348015608d576000080fd5b50609460bc565b60405180828152602001915050604051809103906000f35b816000819055508060018190555050565b60006001546000541215560d0576001546000545b9050905600a165627a7a723058208591e8c5e4e22d38985b8d8c12b2b67790689f3b972c69e7b9163e7f77bd25200029",gas:500000})
lab3>


Terminal
lab3> geth --datadir bkc_data init ~/lab3/genesis.json
INFO [10-15|22:33:22] Maximum peer count ETH=25 LES=0 total=25
INFO [10-15|22:33:22] Allocated cache and file handles database=/home/user1/lab3/bkc_data/geth/chaindata cache=16 handles=16
INFO [10-15|22:33:24] Persisted trie from memory database nodes=0 size=0.00B time=1.935µs gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [10-15|22:33:24] Successfully wrote genesis state database=chaindata hash=b913d0…07d3df
INFO [10-15|22:33:24] Allocated cache and file handles database=/home/user1/lab3/bkc_data/geth/lightchaindata cache=16 handles=16
INFO [10-15|22:33:24] Persisted trie from memory database nodes=0 size=0.00B time=2.053µs gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [10-15|22:33:24] Successfully wrote genesis state database=lightchaindata hash=b913d0…07d3df
lab3> geth --datadir bkc_data --networkid 89992018 --bootnodes enode://d3cd4e70fe7ad1dd7fb23539c53982e42816b4218cc370e8af13945f7b5e2b4a288f8b949dbdba6a998c9141266a0df61523de74490c91fc1e3d538b299bb8ab@128.230.208.73:30301 console 2>console.log
Welcome to the Geth JavaScript console!

instance: Geth/v1.8.9-stable-ff9b1461/linux-amd64/go1.10
coinbase: 0x1a0aeb2893f19e854eabdc4822b51693cd1321c5
at block: 90359 (Sat, 06 Oct 2018 18:28:57 EDT)
 datadir: /home/user1/lab3/bkc_data
 modules: admin:1.0 debug:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

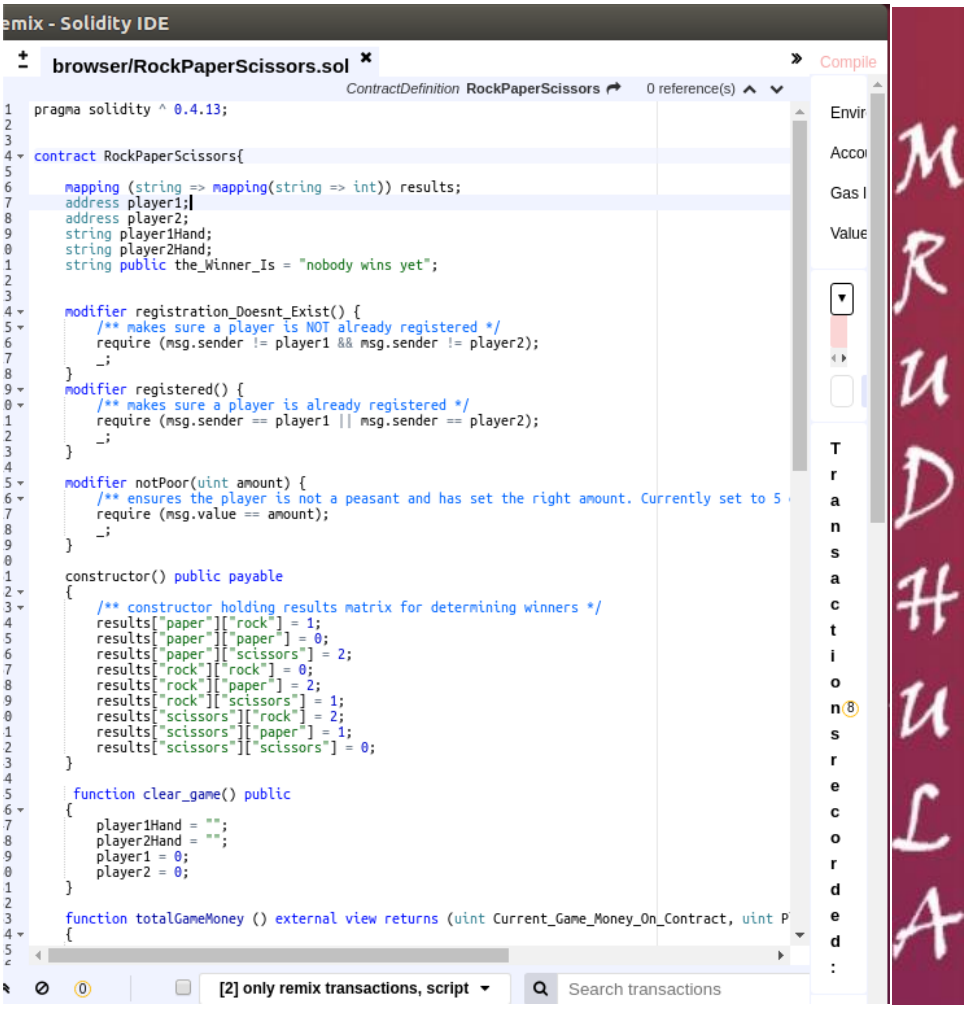
> admin.addPeer("enode://d2547d500b1e982ac93a6ce1dbf34cff6545987740313373ccecb28e095c6ce4294e5cf4be2f002672d30fb717b8bd05e1a12163b24743b907bb7d2c37415928@[128.230.208.73]:30303")
true

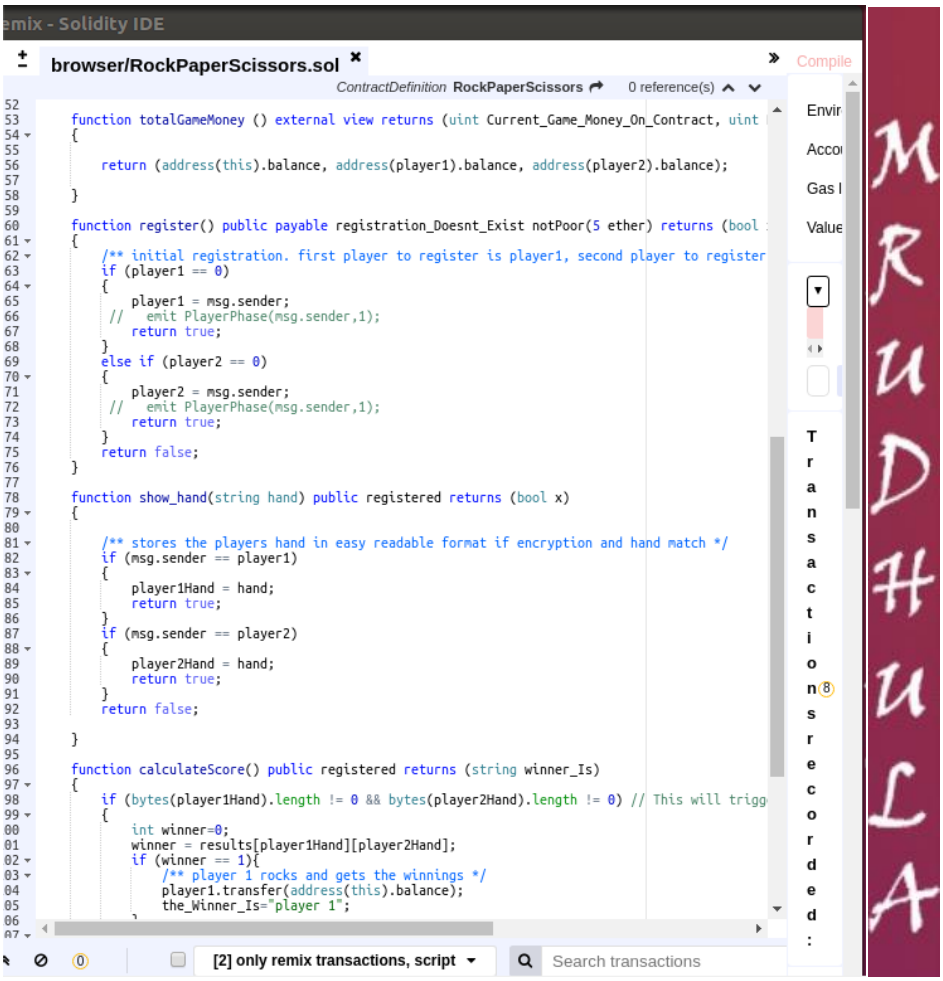


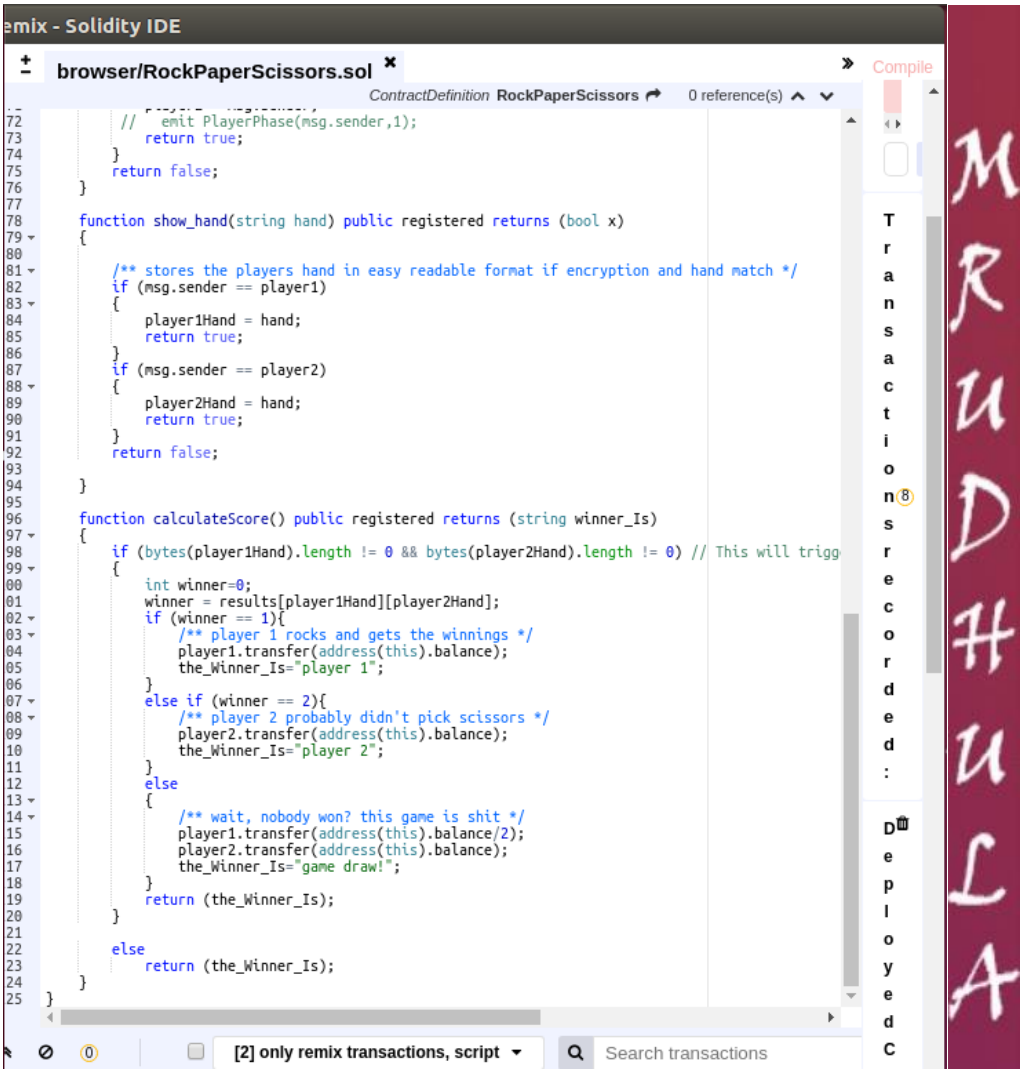
But here in send transaction we 4 parameters whereas in hello-world program we had given three. Here we give two integer numbers that are to be compared to find maximum value, the account detail from and gas value.

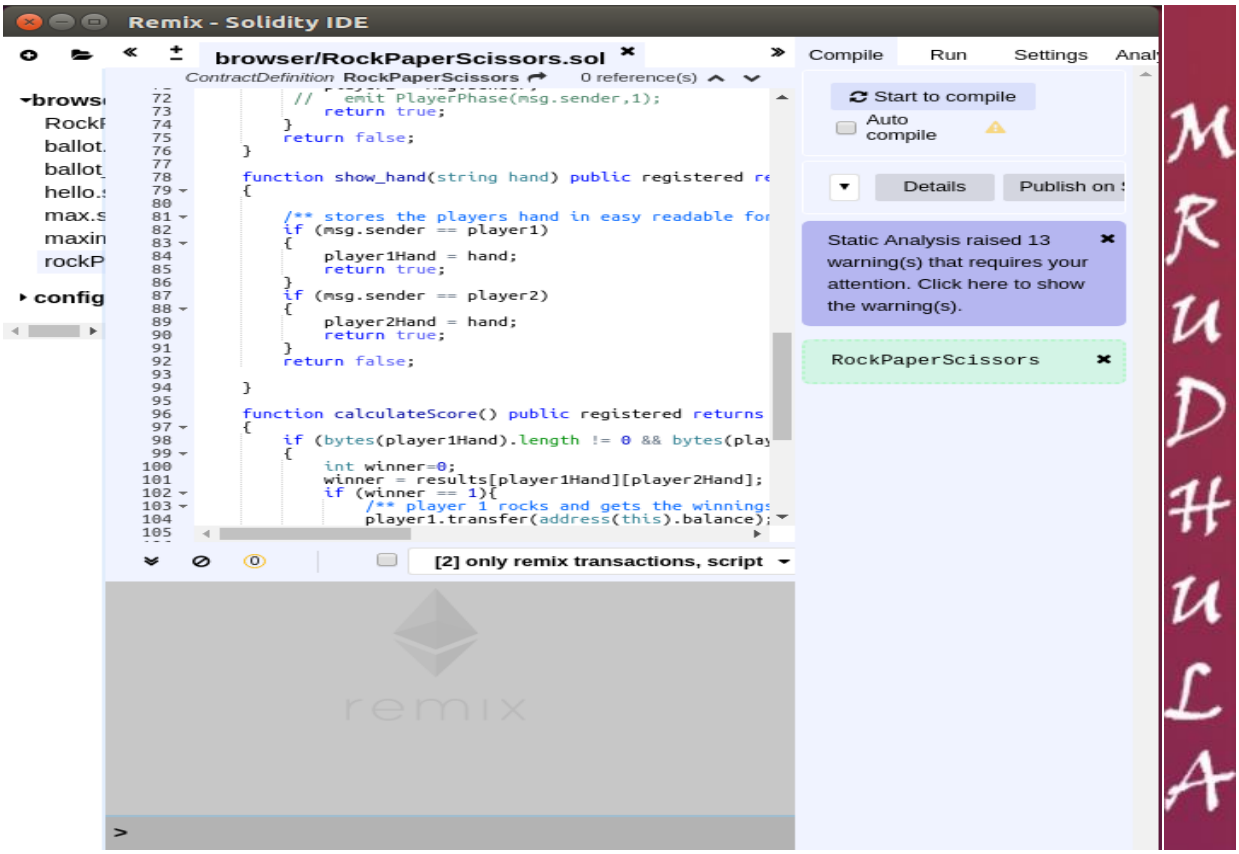
EXERCISE 2: Rock-paper-scissors game

Below is the code for the program rock paper scissors game.

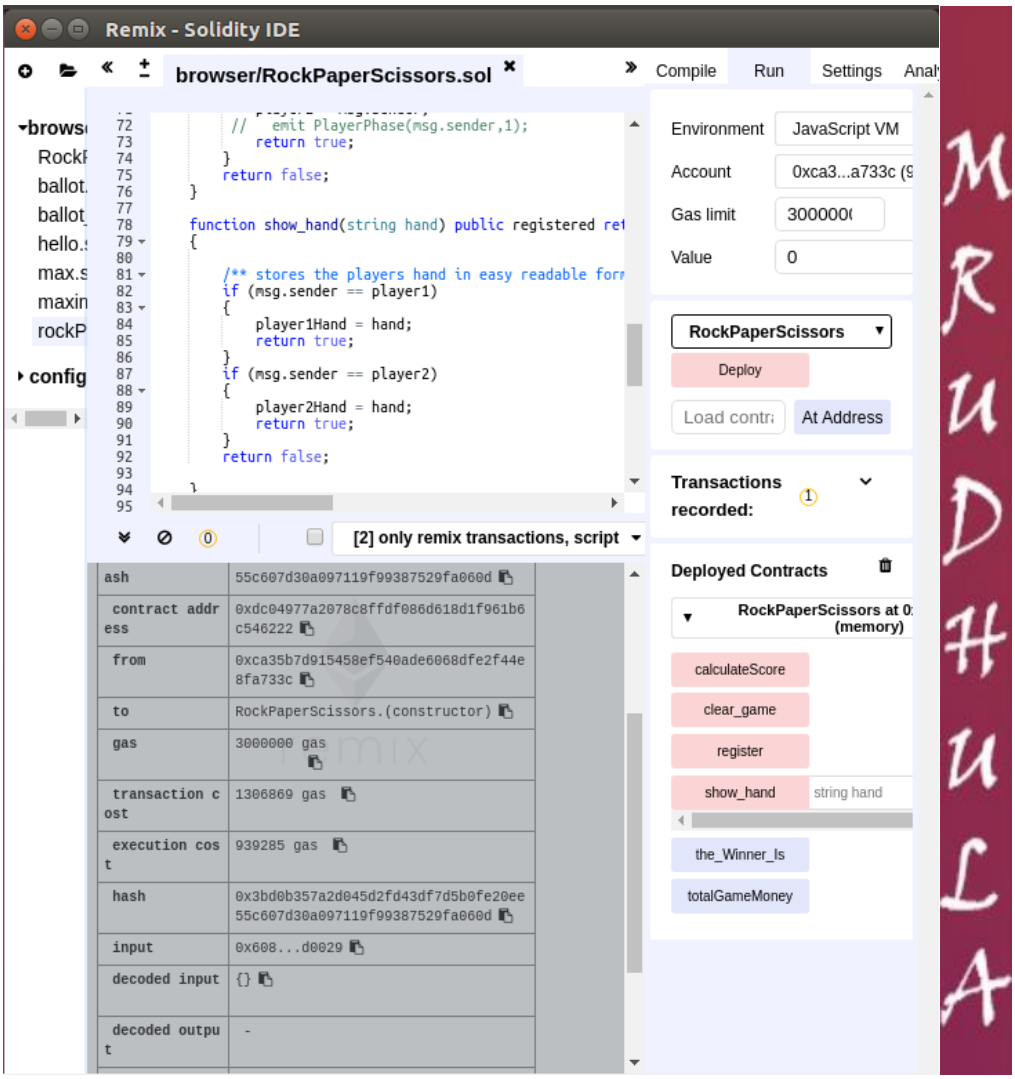



emix - Solidity IDE
browser/RockPaperScissors.sol
Compile
ContractDefinition RockPaperScissors
0 reference(s)
function totalGameMoney () external view returns (uint Current_Game_Money_On_Contract, uint
{
return (address(this).balance, address(player1).balance, address(player2).balance);
}
function register() public payable registration_Doesnt_Exist notPoor(5 ether) returns (bool
{
/** initial registration. first player to register is player1, second player to register
if (player1 == 0)
{
player1 = msg.sender;
// emit PlayerPhase(msg.sender,1);
return true;
}
else if (player2 == 0)
{
player2 = msg.sender;
// emit PlayerPhase(msg.sender,1);
return true;
}
return false;
}
function show_hand(string hand) public registered returns (bool x)
{
/** stores the players hand in easy readable format if encryption and hand match */
if (msg.sender == player1)
{
player1Hand = hand;
return true;
}
if (msg.sender == player2)
{
player2Hand = hand;
return true;
}
return false;
}
function calculateScore() public registered returns (string winner_Is)
{
if (bytes(player1Hand).length != 0 && bytes(player2Hand).length != 0) // This will trigg
{
int winner=0;
winner = results[player1Hand][player2Hand];
if (winner == 1){
/** player 1 rocks and gets the winnings */
player1.transfer(address(this).balance);
the_Winner_Is="player 1";
Envir
Accou
Gas l
Value
Transactions recorded:
[2] only remix transactions, script
Search transactions
MRUDHULA

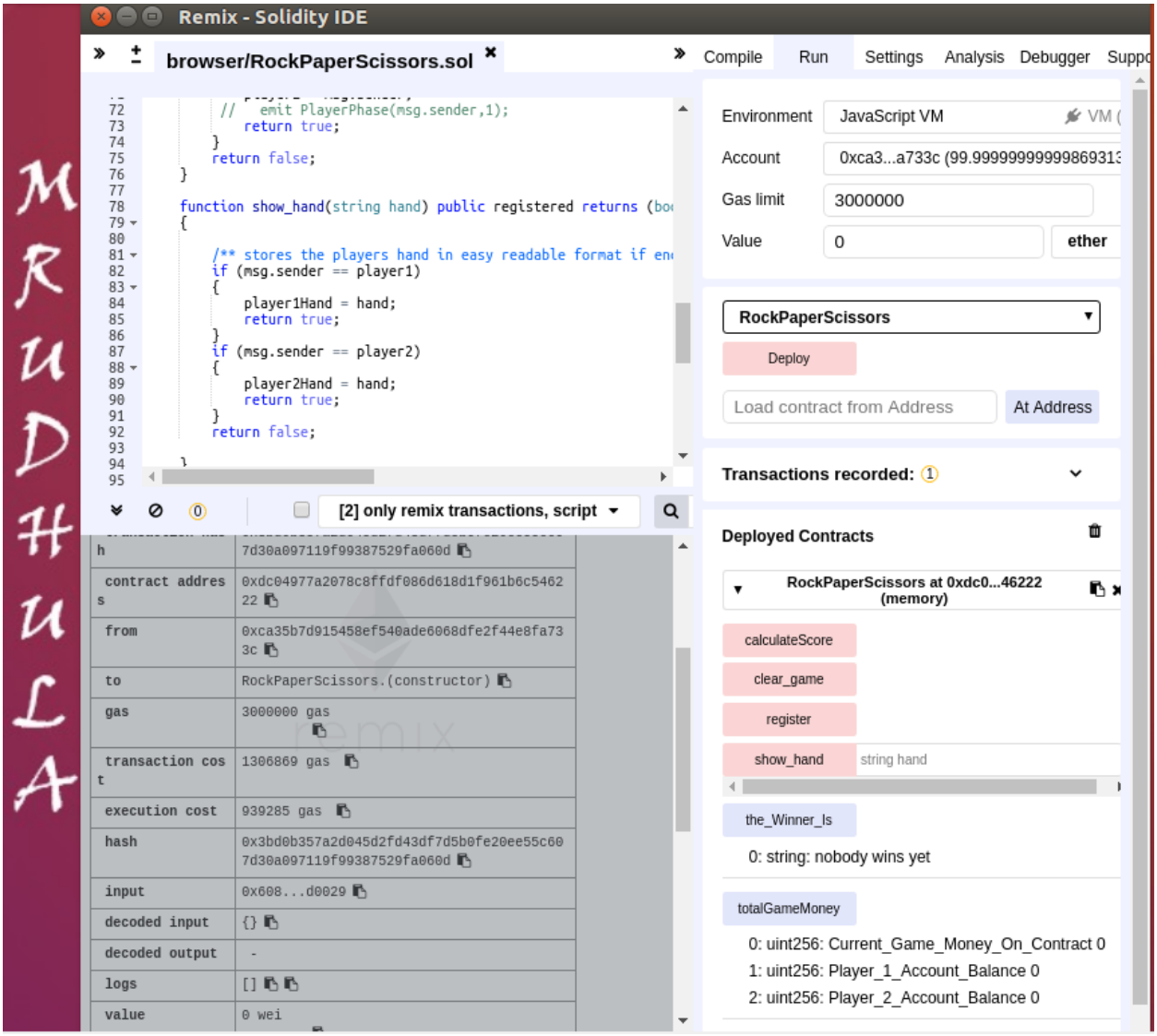

emix - Solidity IDE
browser/RockPaperScissors.sol
ContractDefinition RockPaperScissors
0 reference(s)
Compile
// emit PlayerPhase(msg.sender,1);
return true;
}
return false;
}
function show_hand(string hand) public registered returns (bool x)
{
/** stores the players hand in easy readable format if encryption and hand match */
if (msg.sender == player1)
{
player1Hand = hand;
return true;
}
if (msg.sender == player2)
{
player2Hand = hand;
return true;
}
return false;
}
function calculateScore() public registered returns (string winner_Is)
{
if (bytes(player1Hand).length != 0 && bytes(player2Hand).length != 0) // This will trigg
{
int winner=0;
winner = results[player1Hand][player2Hand];
if (winner == 1){
/** player 1 rocks and gets the winnings */
player1.transfer(address(this).balance);
the_Winner_Is="player 1";
}
else if (winner == 2){
/** player 2 probably didn't pick scissors */
player2.transfer(address(this).balance);
the_Winner_Is="player 2";
}
else
{
/** wait, nobody won? this game is shit */
player1.transfer(address(this).balance/2);
player2.transfer(address(this).balance);
the_Winner_Is="game draw!";
}
return (the_Winner_Is);
}
else
return (the_Winner_Is);
}
}
Transactions recorded:
Deployed C
[2] only remix transactions, script
Search transactions
MRUDHULA



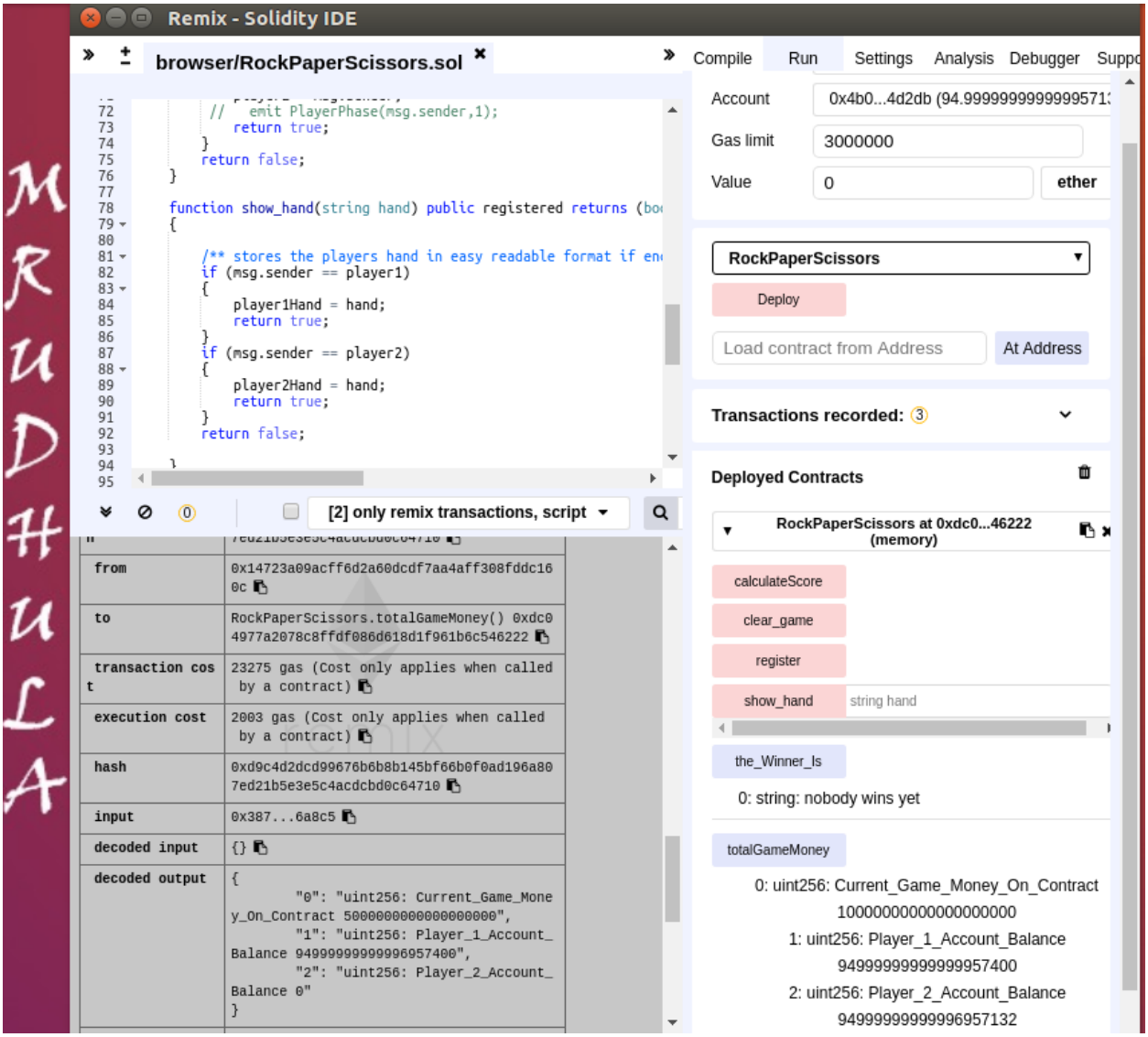
We write the code for rock paper scissors game where two players will play the game and initially we set a limit for ethers to be bid by each player which they register with and then show their hands and then we calculate the winner and who the ether goes to. We compile the program.



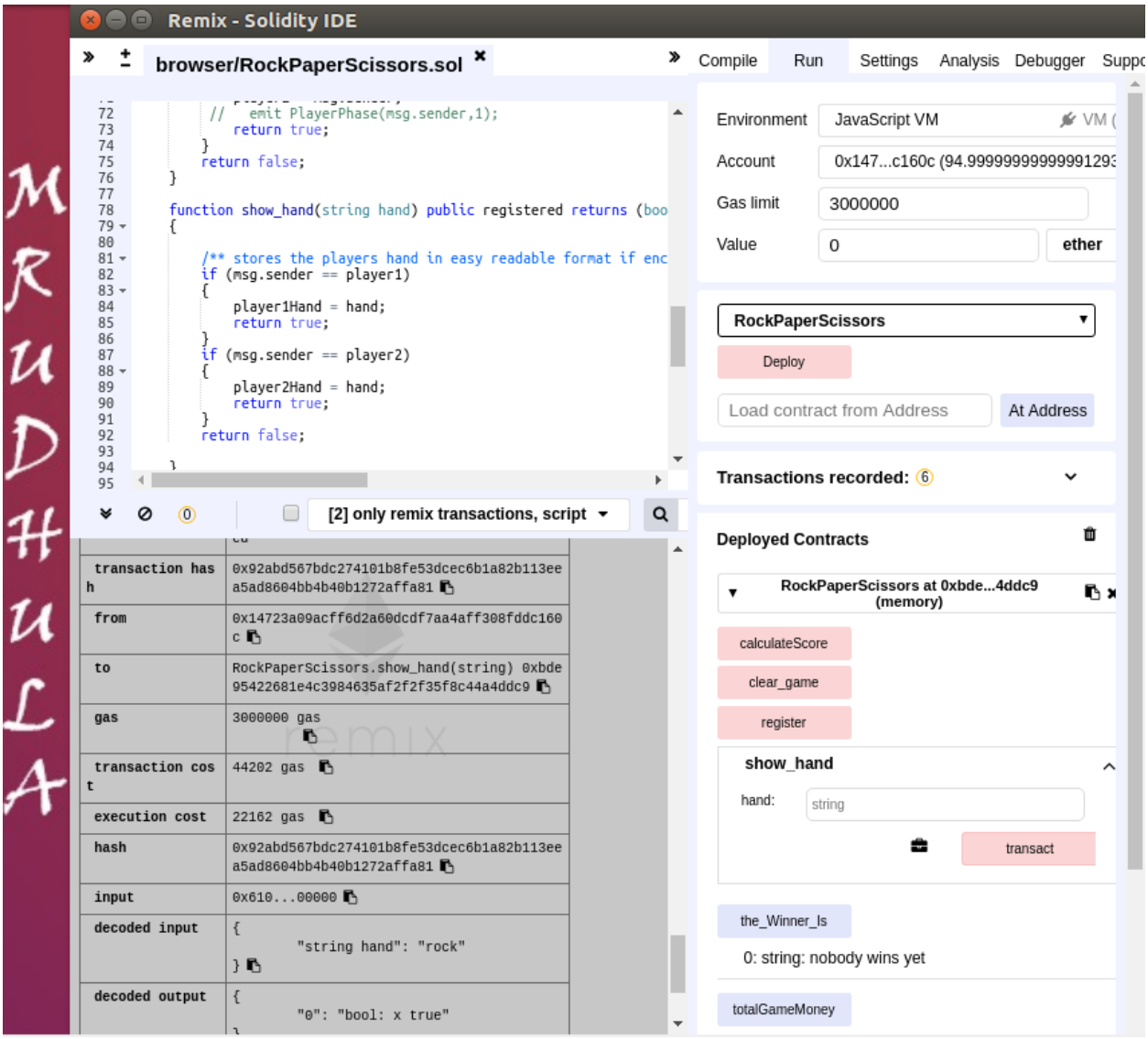
We run the program and click on deploy and see in console log that currently no decoded input or output is assigned.



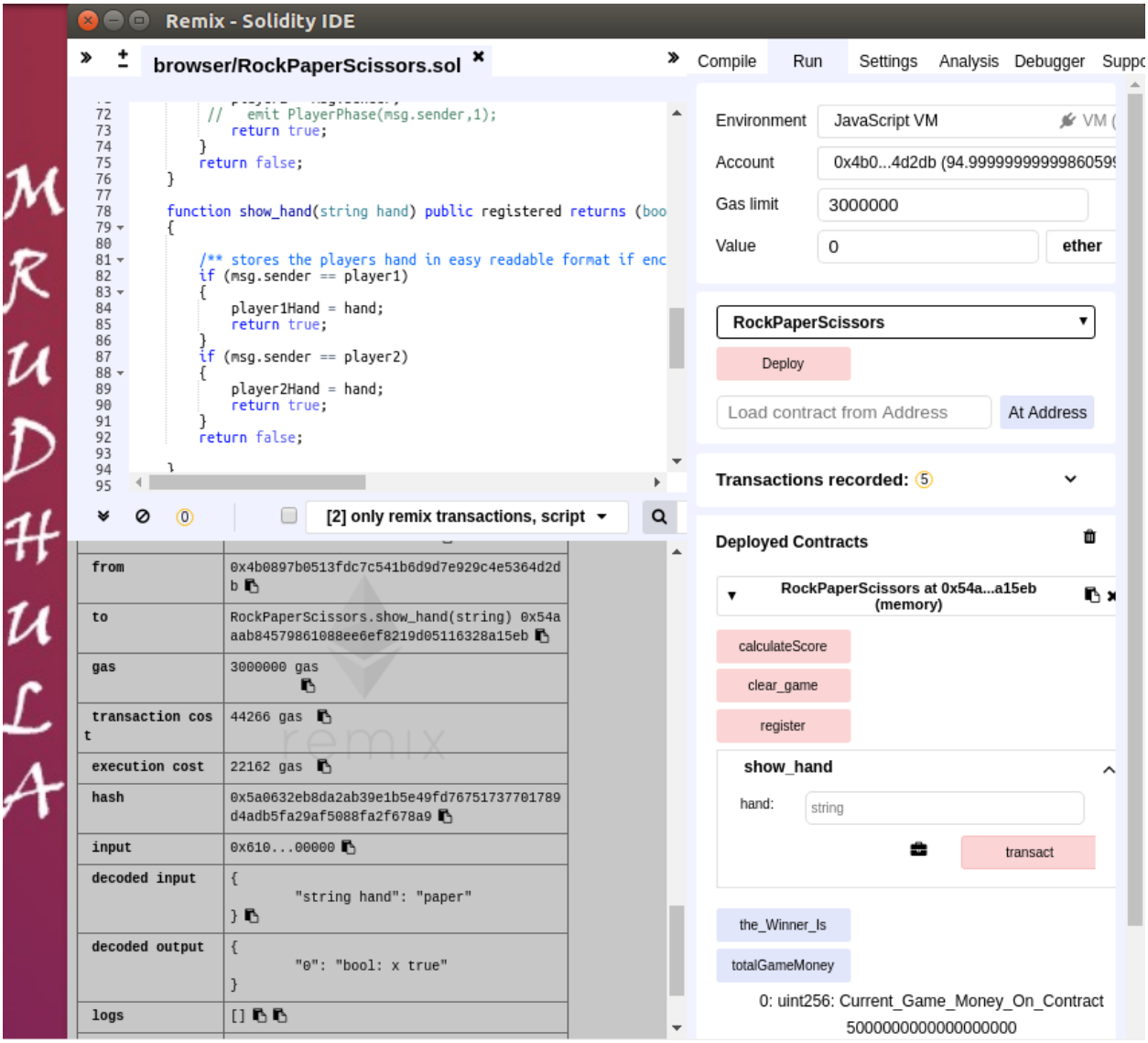
We then check the totalGameMoney function, this function will show how many wei is left after each player has bid and how much ether the contract has. Currently we see that there are no transactions. Hence, balance is 0 throughout.



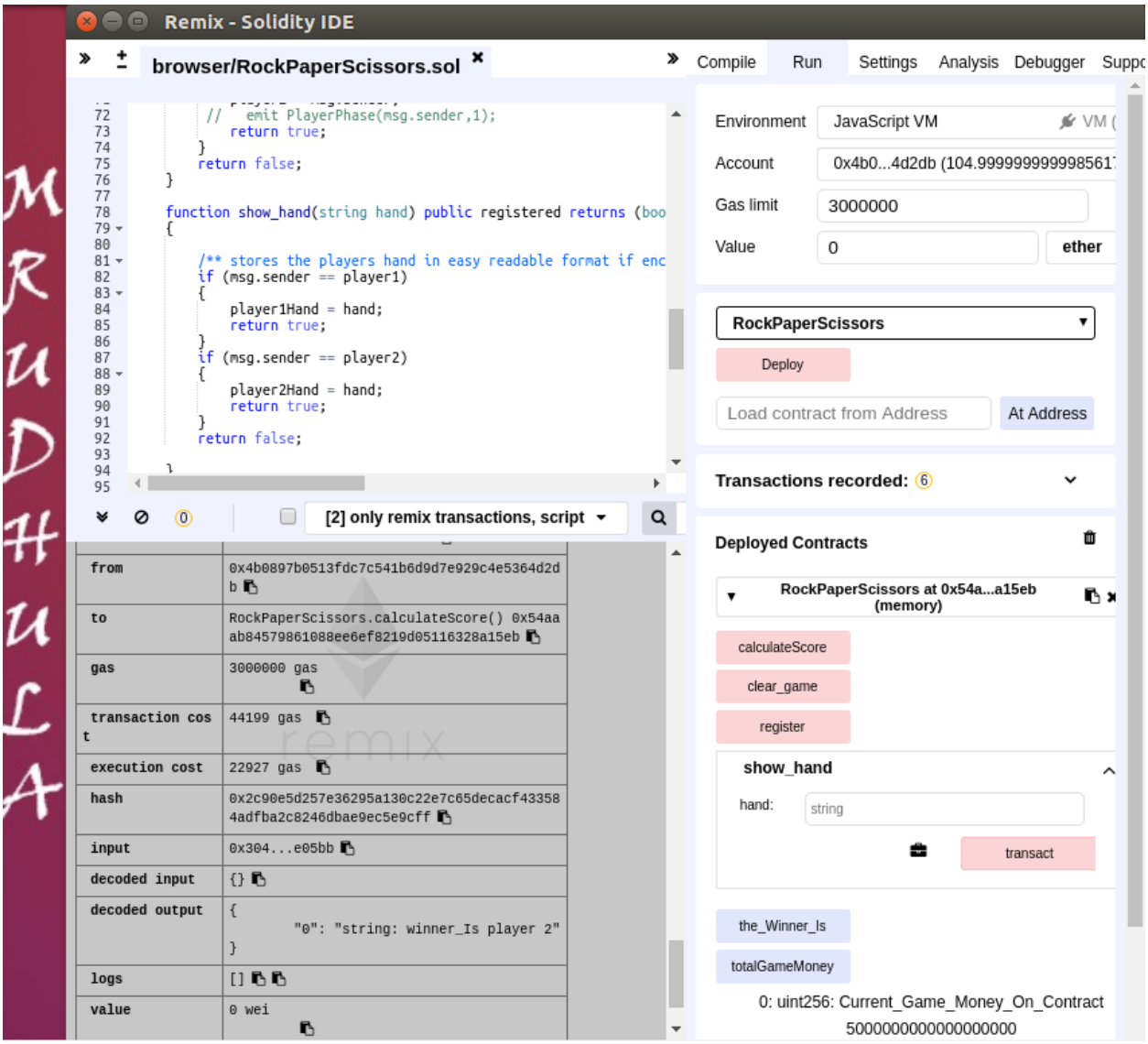
Then we register each player with 5 ether each. And then we check the balance we see that 5 ethers have been reduced from 100 ethers and the balance is shown. Currently contract will have 10 ethers, 5 each from the two players playing.



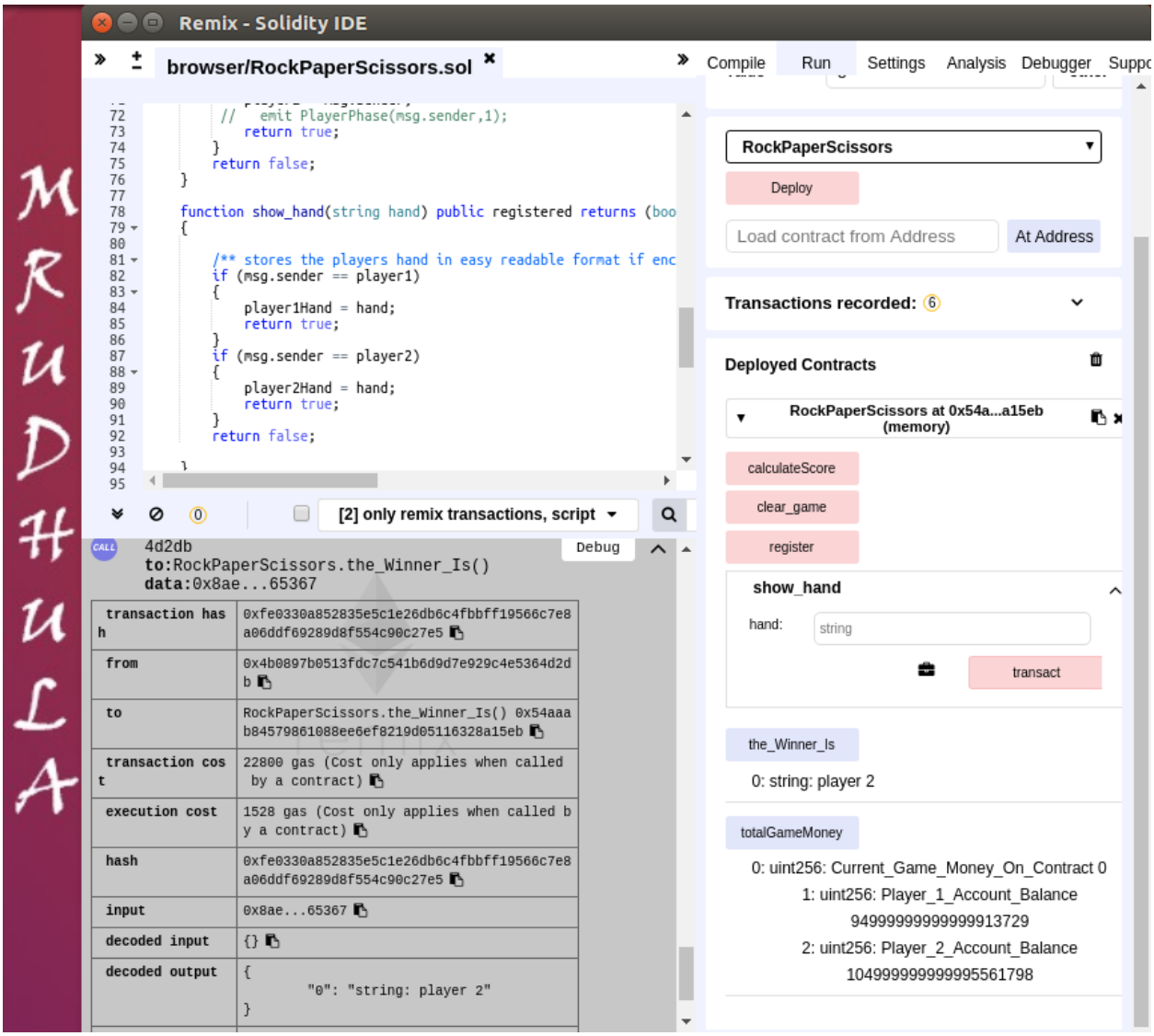
We initiate the hand for player 1 as shown in the above screenshot player 1 plays rock as shown in console log above and we click transact for the hand to get stored. When we check for the winner it says there are no winners yet.



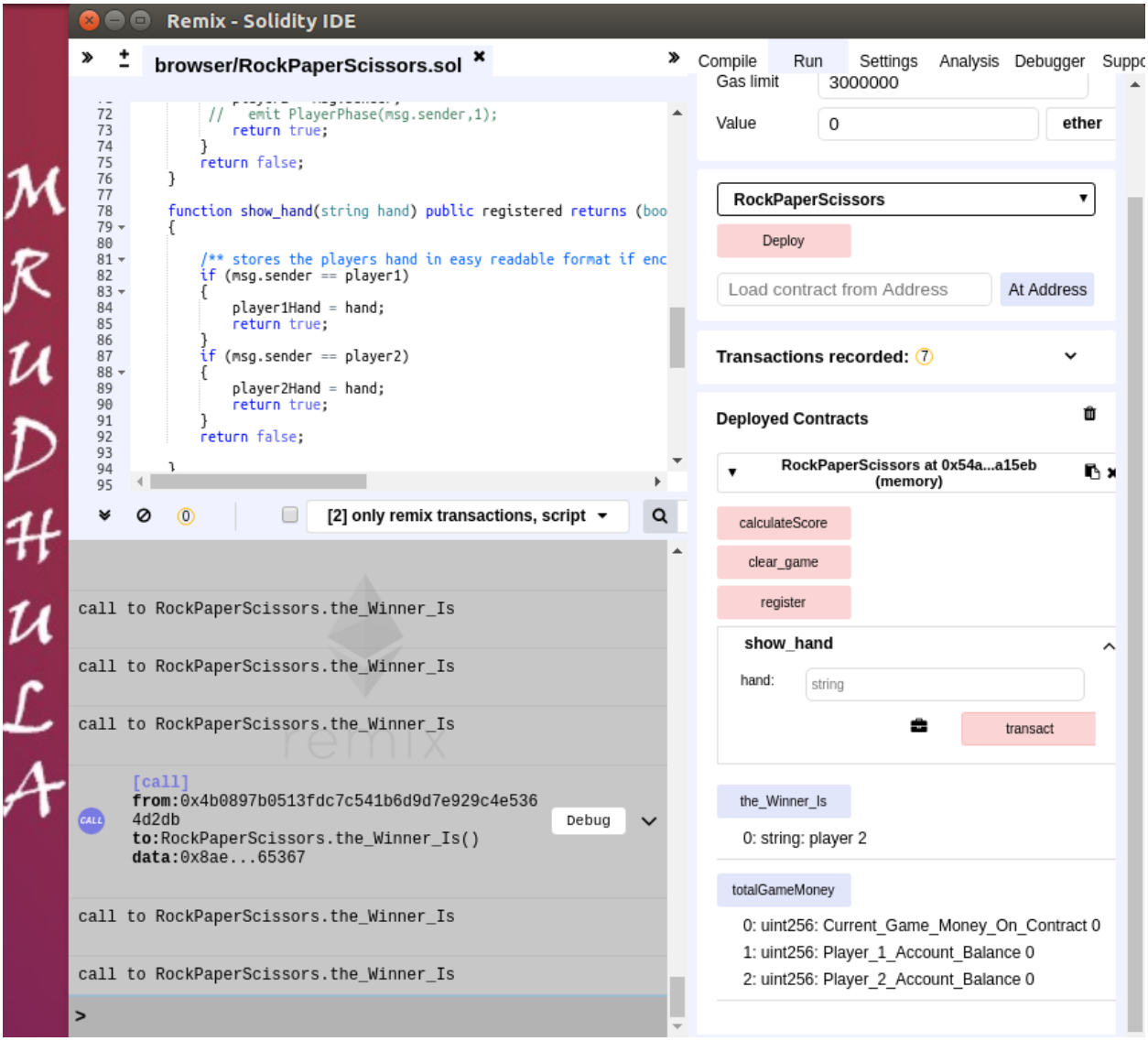
Then we initiate the hand for player 2. Player 2 plays paper as shown in console log. Then we click on transact to store player 2's hand.



Then we click on calculate score this function calculates from the matrix and checks which condition either players are in. when we check the console log for the calculateScore we can see the winner is specified in decoded output.



Now when we click on the totalGameMoney function we see that the winner gets all the ether that contract had. The balance against the contract is 0 ethers, against player 1 is still the total – 5 ethers and players 2 gets 10 complete ethers.



We have another option that is to clear game when we click on it the balance gets cleared out and zero's out as shown above.