Dhanashree Srinivasa

SUID: 393473169

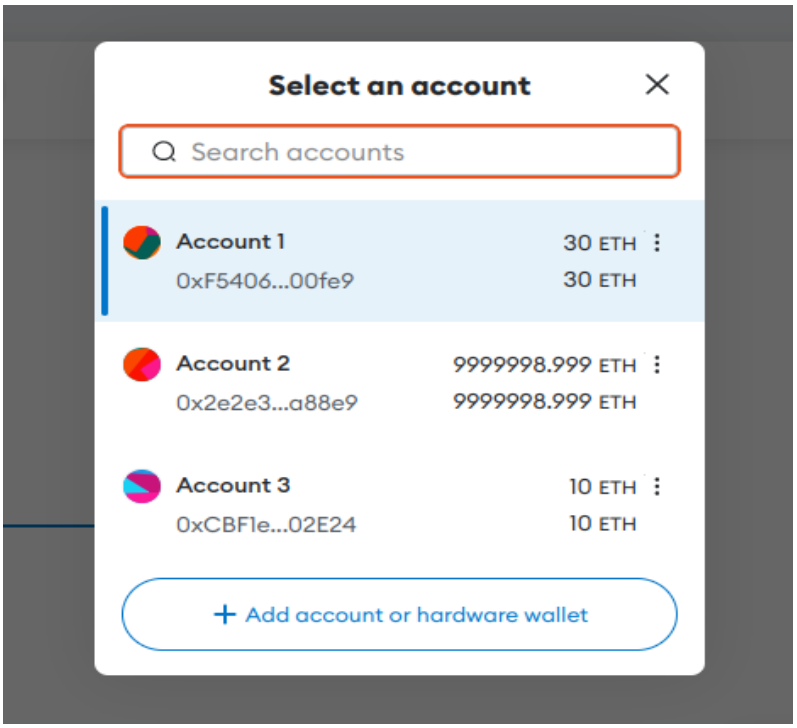
Course: Computer Security - CSE 643

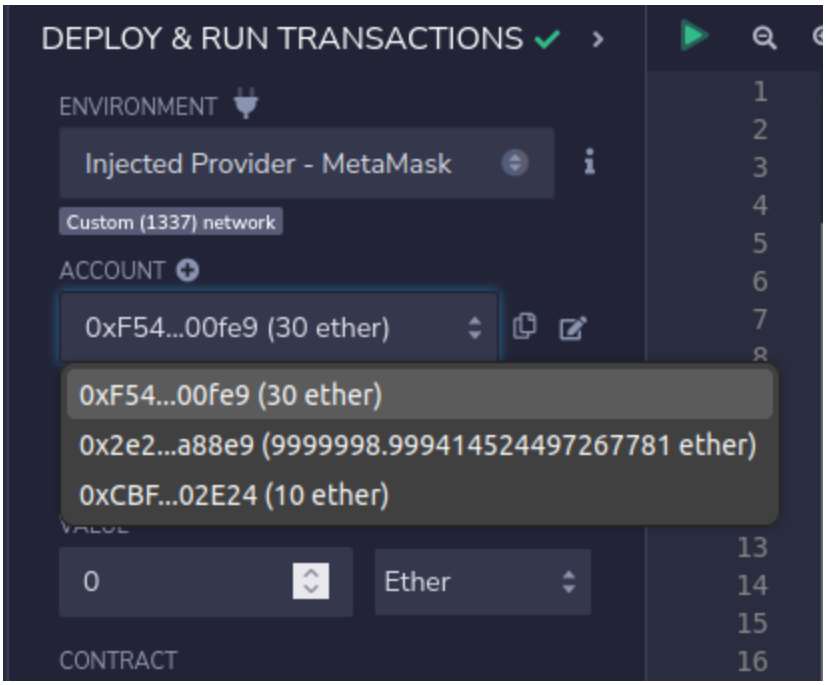
Smart Contract Lab

Task 1: Using Remix for Smart Contract Development

Task 1.a: Connecting Remix to the SEED Emulator

Connect Remix with Metamask SEED Emulator.

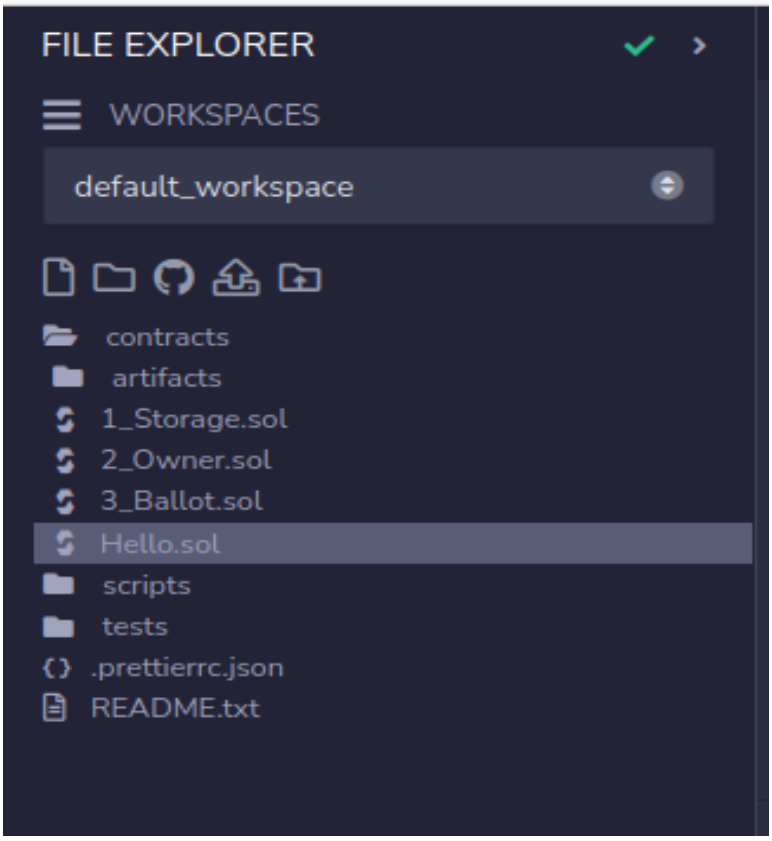




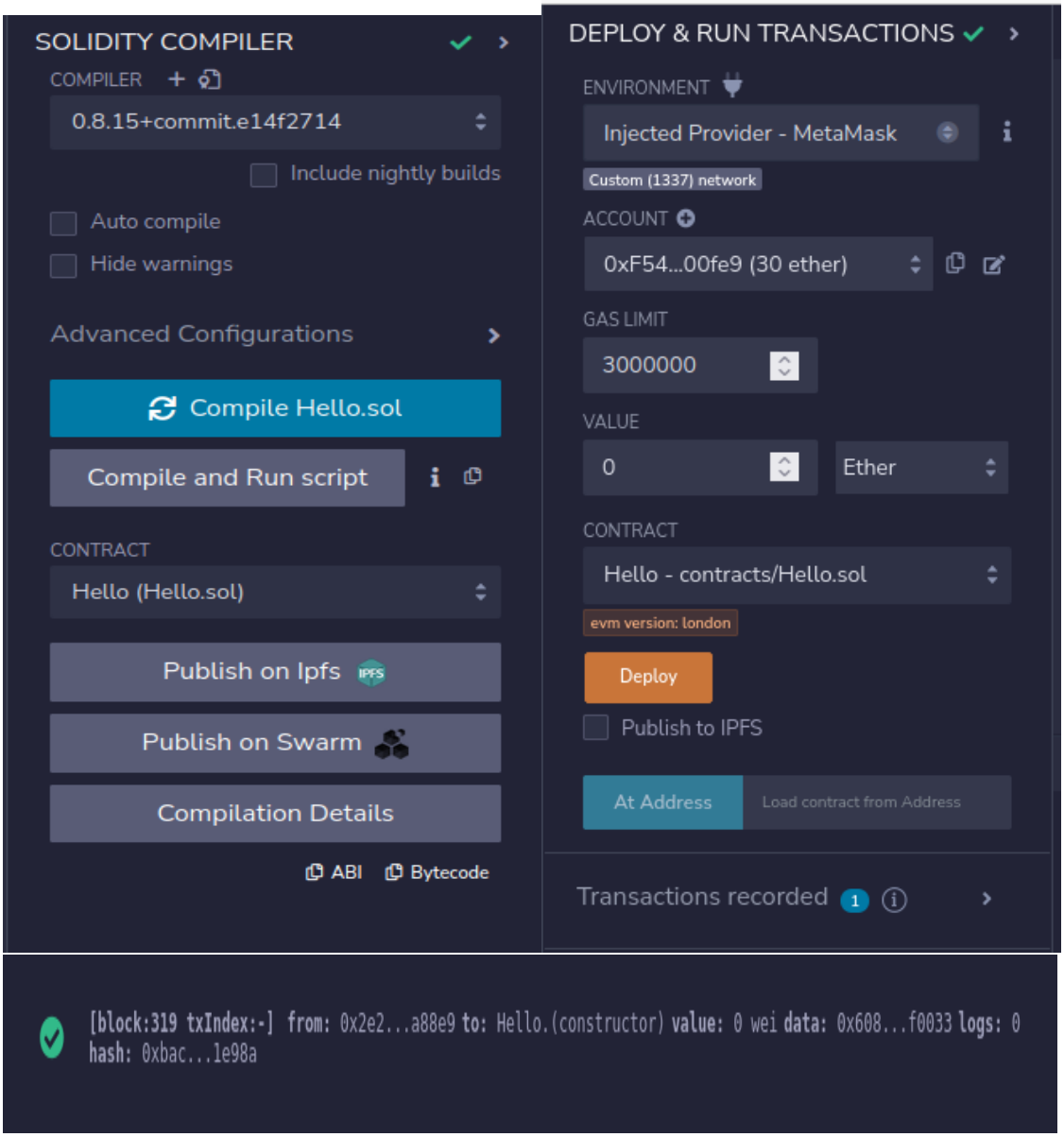
The above screenshot shows Metamask connected to Remix. All 3 accounts are displayed on Remix.

1.b: Write, Compile, and Deploy Smart Contract

Add the code Hello.sol under the contracts folder, compile and deploy.


FILE EXPLORER
WORKSPACES
default_workspace
contracts
artifacts
1_Storage.sol
2_Owner.sol
3_Ballot.sol
Hello.sol
scripts
tests
.prettierrc.json
README.txt

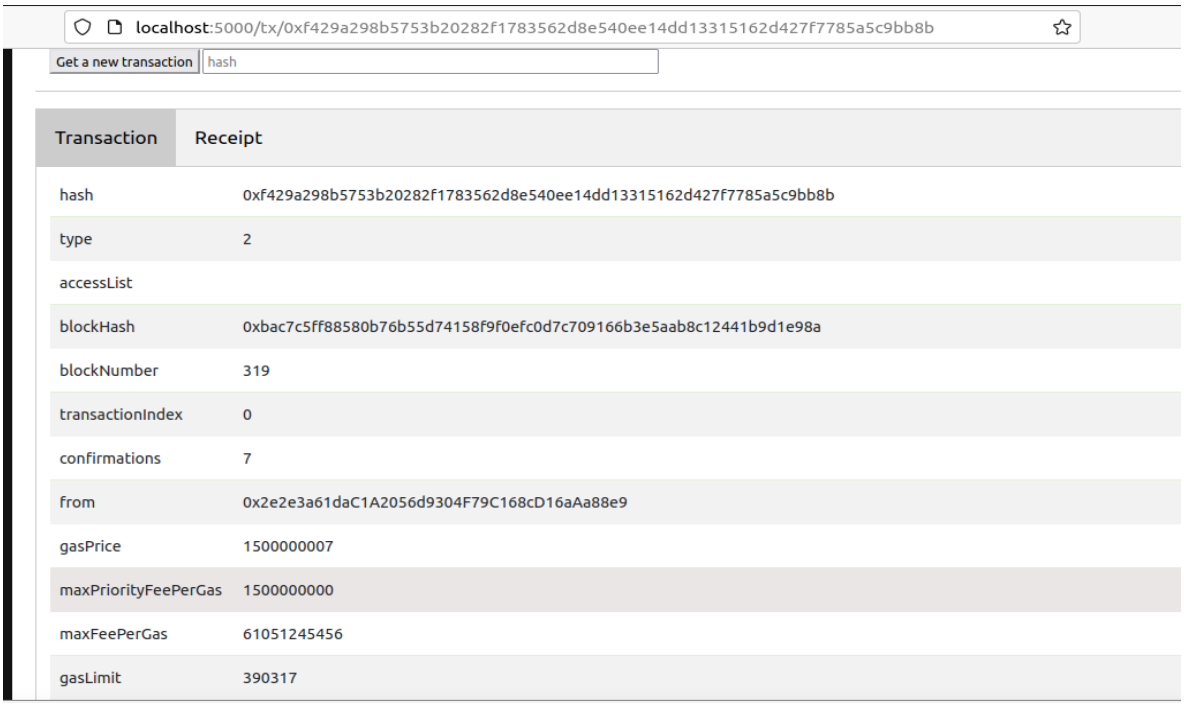

Home
Hello.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.15;
contract Hello{
address public owner;
uint256 counter;
constructor(){ 315997 gas 291400 gas
owner = msg.sender;
}
receive() external payable { } undefined gas
function sayHello() public pure returns (string memory) { infinite gas
return "Hello World";
}
function getResult(uint a, uint b) public view returns (uint) { infinite gas
uint sum = a + b + counter;
return sum;
}



As shown above, the Hello.sol contract is compiled and deployed.

1.c: Under the hood

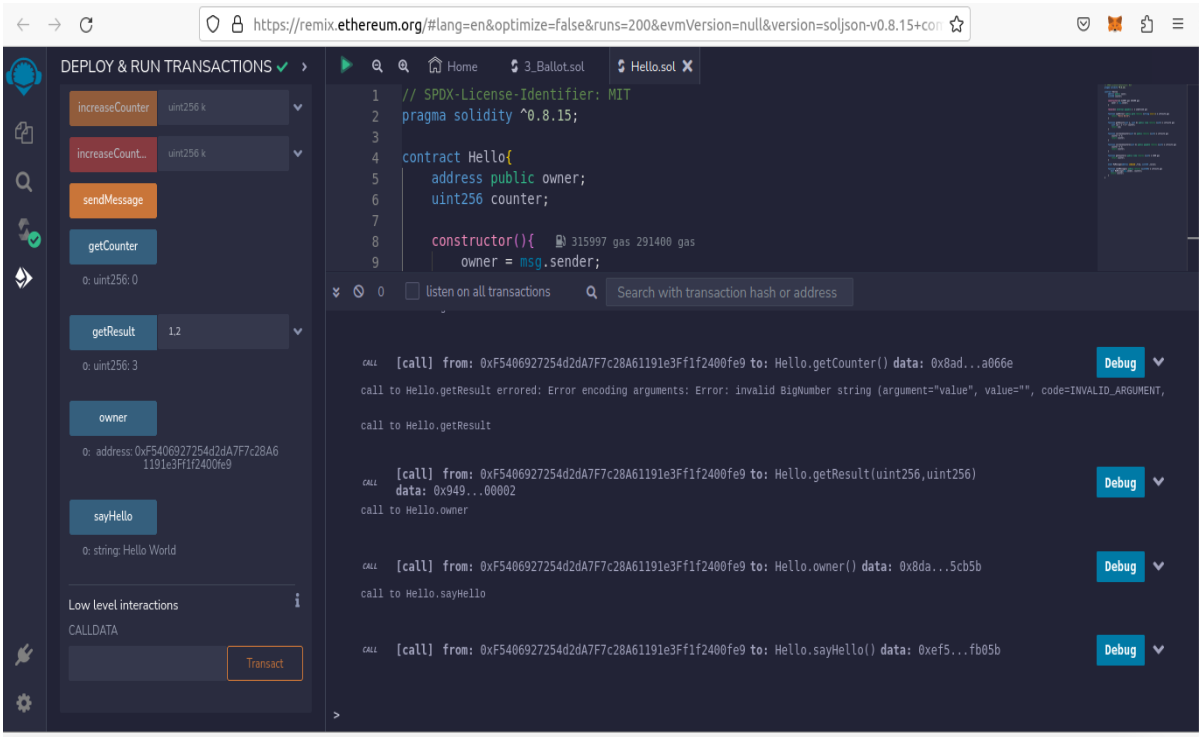
There is a block created for each transaction. When Ether is transferred from account 2 to account 1 the recipient address is account 1. But when Hello.sol is deployed the recipient address is null.
Can be observed in the screenshot below. And also, in the contract deployment the data field has some value.



Task 2: Invoke Contract Functions

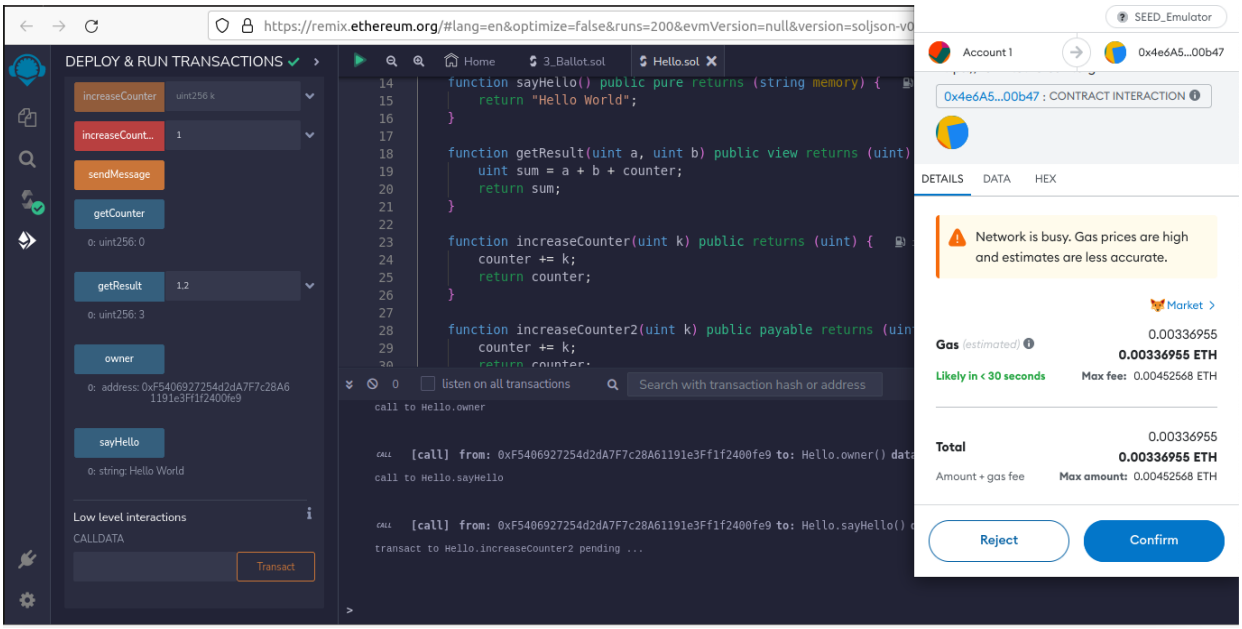
Task 2.a: Invoke a function via local call

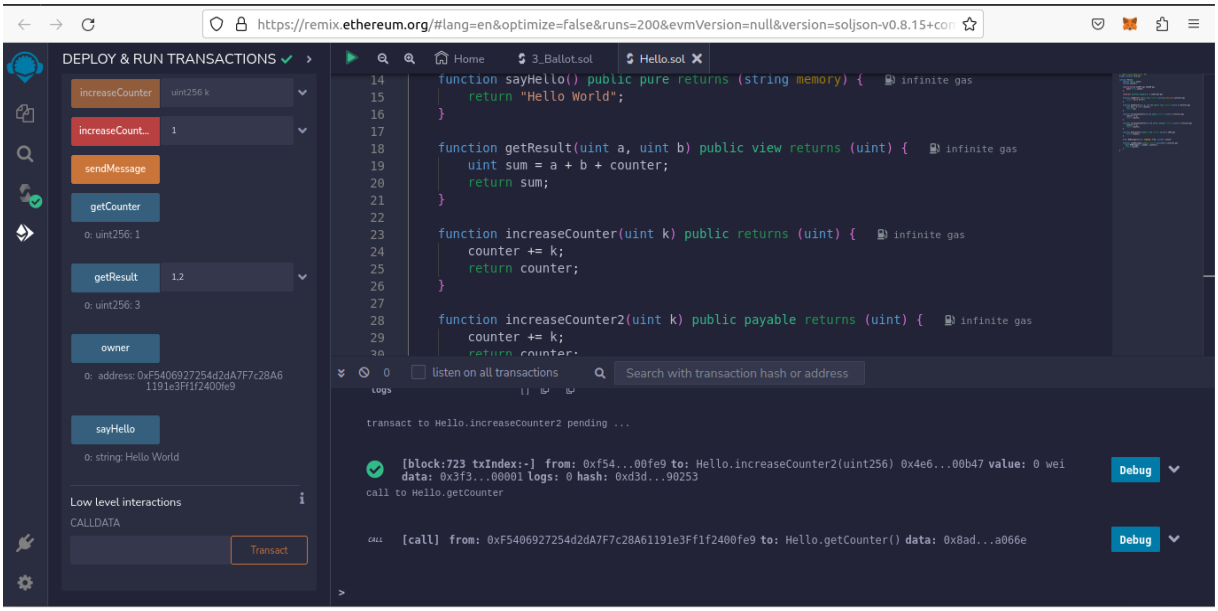
The below screenshot shows that all the public view type functions are executed via local calls.



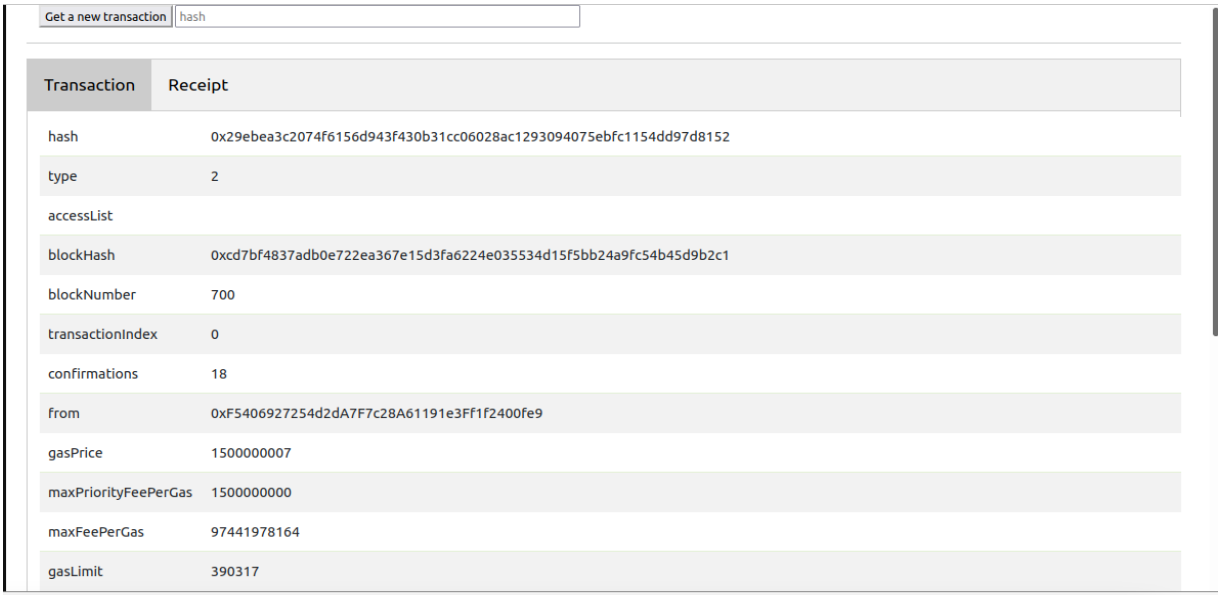
Task 2.b: Invoke a function via transaction

The below screenshot shows that increaseCounter() was executed on Remix and the counter value was increased by 1.





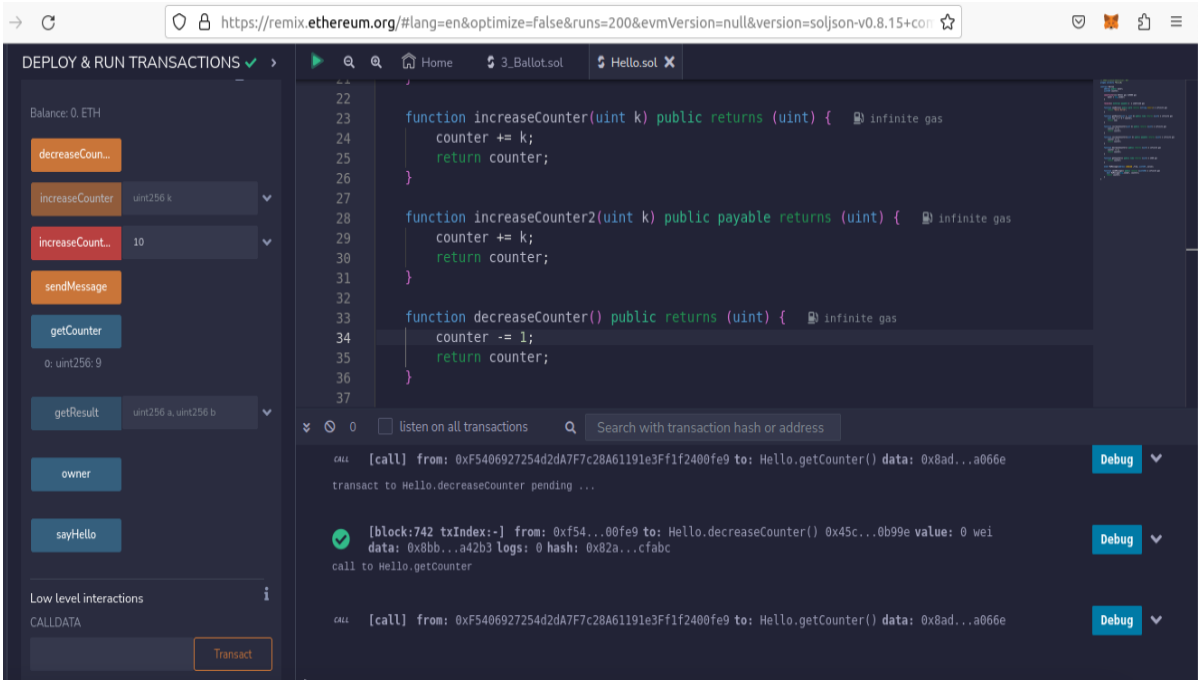
The below screenshot shows the block which was executed when increaseCounter() call was invoked.



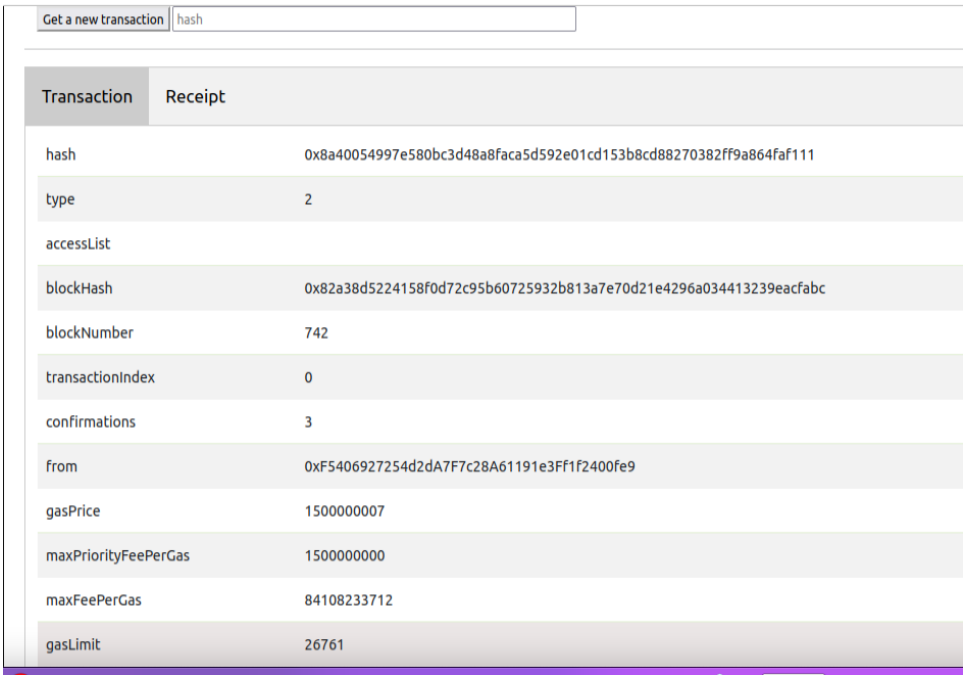
Create a function “decreaseCounter” in Hello.sol, which decrease the counter value by 1.

```
function decreaseCounter() public returns (uint) {    infinite gas
    counter -= 1;
    return counter;
}
```

I have initially increased the counter value by 10 using the increaseCounter() and then decreased the counter value by 1. Hence the getCounter() returns 9.

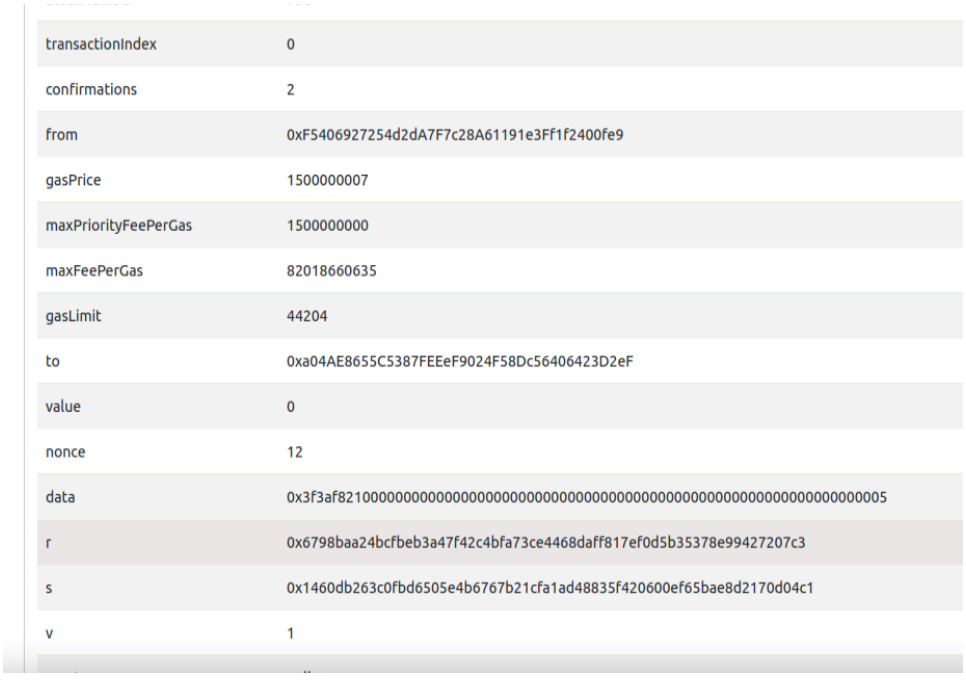


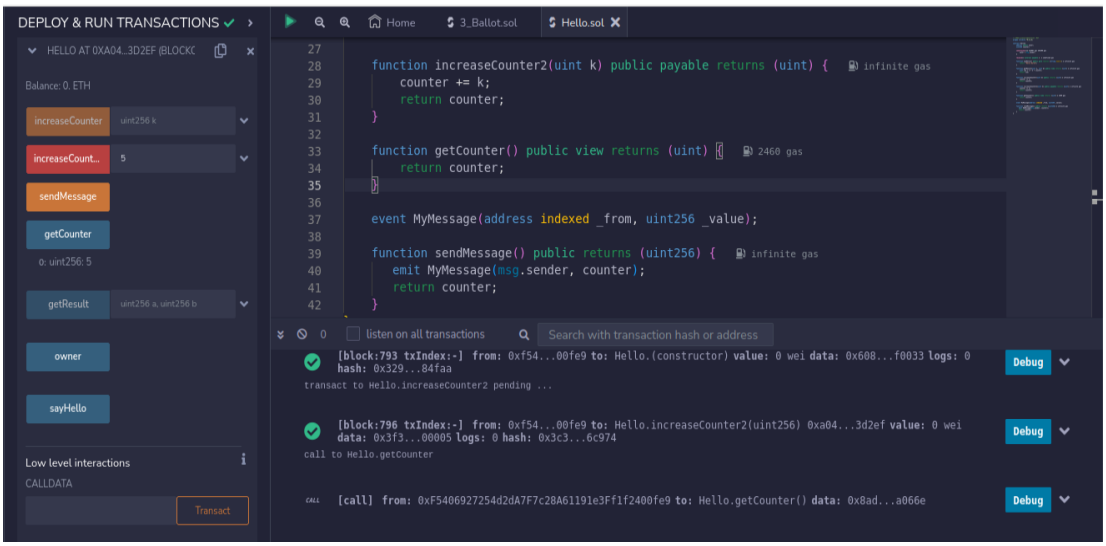
Beow is the block that was returned, after executing the decreaseCounter() function.



Task 2.c: Under the hood

Below is the block that was generated while using increaseCounter() function to 5.





Below is the code used to know the hx value of function prototype.

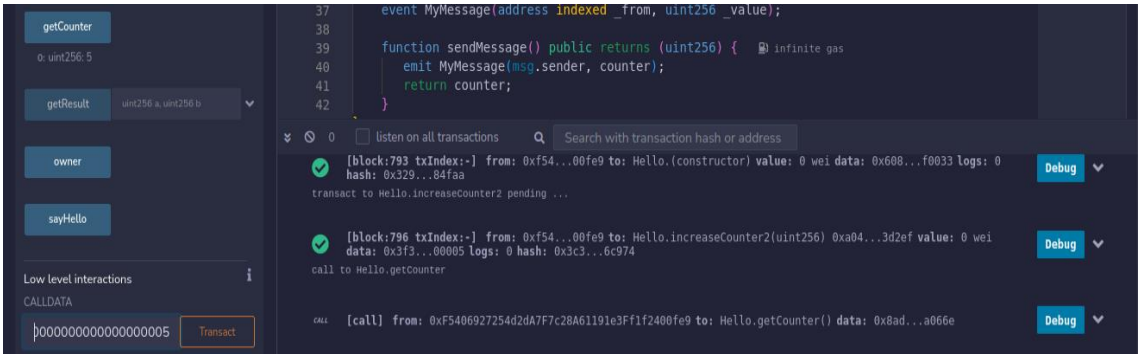
```
>>> from web3 import Web3
>>> hash = Web3.sha3(text="<function signature>")
>>> print(hash.hex())
0x59d8401dc675ba1e957689e93c1c5f917f8ba1f72358193406be595b559d8fed
>>> hash = Web3.sha3(text="increaseCounter2(uint256)")
>>> print(hash.hex())
0x3f3af821d5525a653a7d25f0a1e07cbbd575e6e4a758666d5d2a0c6143dde75a
```

transactionIndex	0
confirmations	60
from	0xF5406927254d2dA7F7c28A61191e3Ff1f2400fe9
gasPrice	1500000007
maxPriorityFeePerGas	1500000000
maxFeePerGas	82018660635
gasLimit	44204
to	0xa04AE8655C5387FEEeF9024F58Dc56406423D2eF
value	0
nonce	12
data	0x3f3af821000000000000000000000000000000000000000000000000000000000000000000000005
r	0x6798baa24bcfbeb3a47f42c4bfa73ce4468daff817ef0d5b35378e99427207c3
s	0x1460db263c0fbd6505e4b6767b21cfa1ad48835f420600ef65bae8d2170d04c1
v	1

The output of the python code can be seen that the first 4 digits are same as the data part should in the block transaction.

data = 0x3f3af821000000000000000000000000000000000000000000000000000000000000000000000005

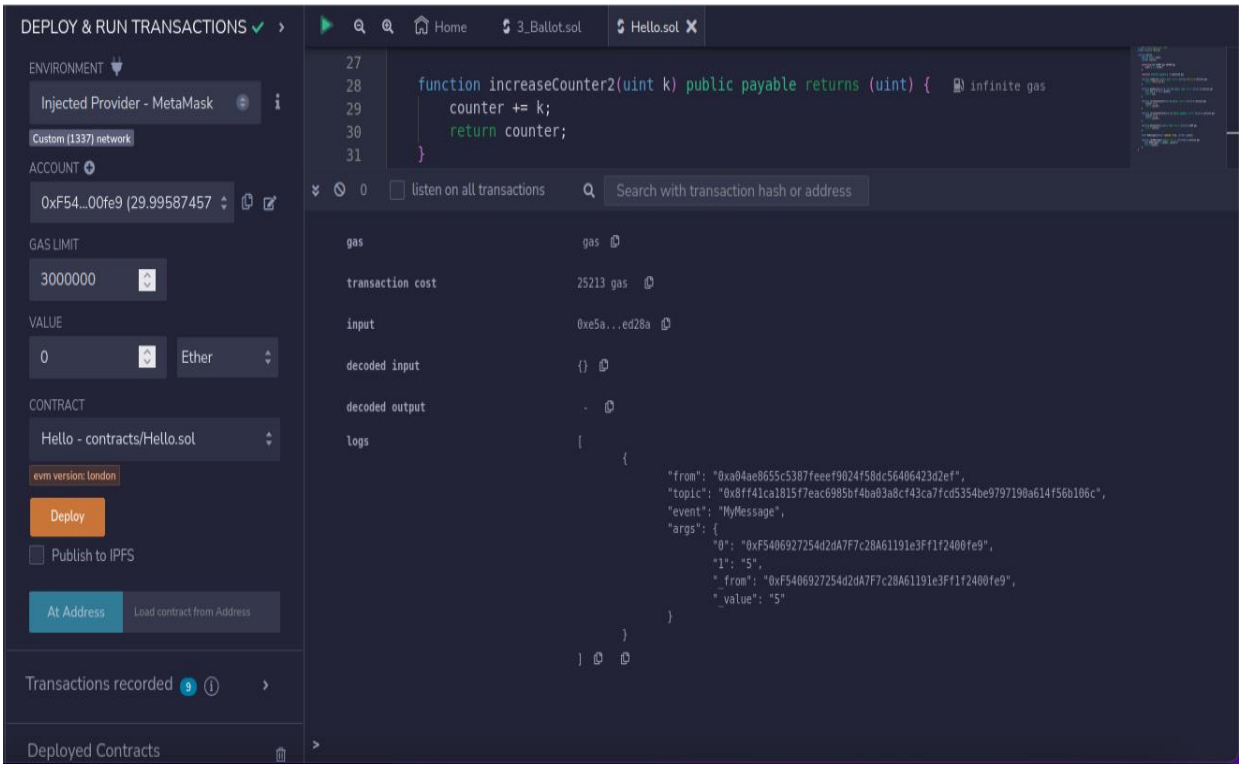
The below screenshot shows that the increaseCounter()'s value is increased 5 times.



Task 2.d: Emit events

Below screenshot shows the output when sendMessage() is executed. The account f54...00fe9 is used.

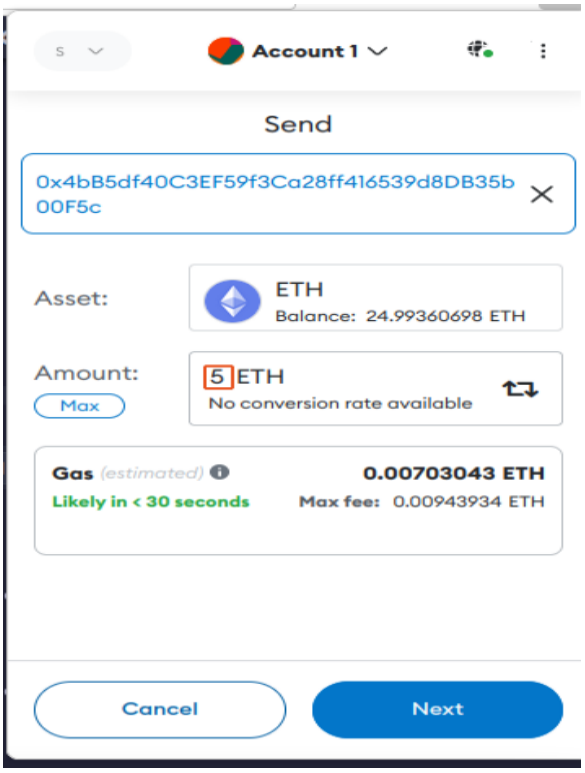
As you can see in the log field shows the “message Sender” and the “value of counter 5”.

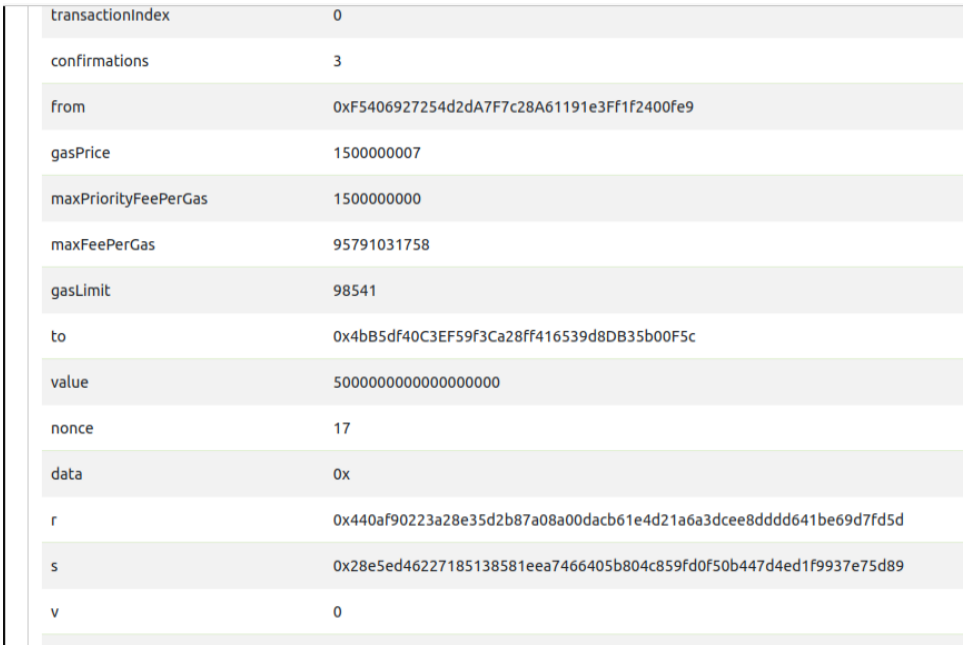


Task 3: Send Fund to Contract

Task 3.a: Send fund directly to a contract address

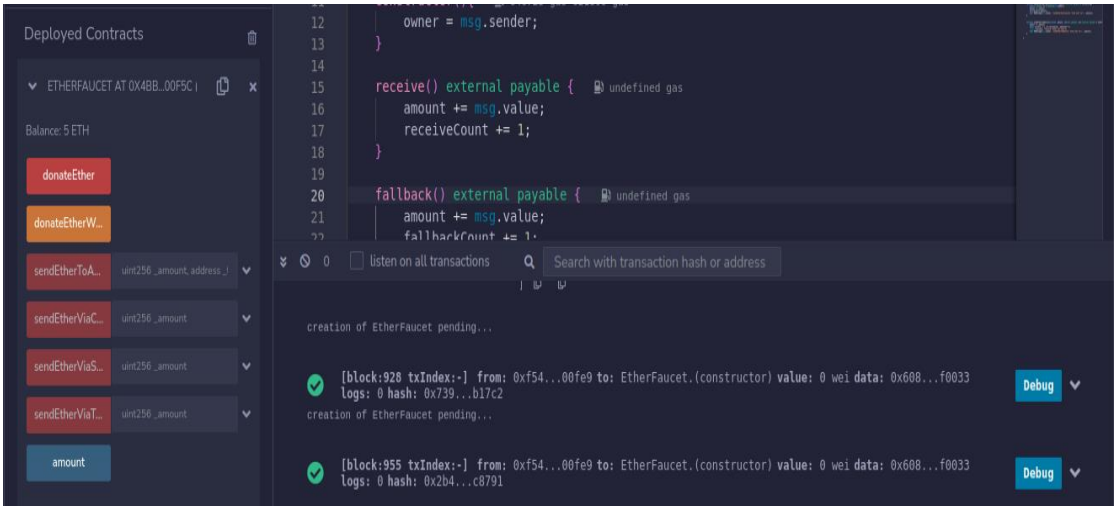
Copy the address of the contract from remix and try to send money to the contract address directly.



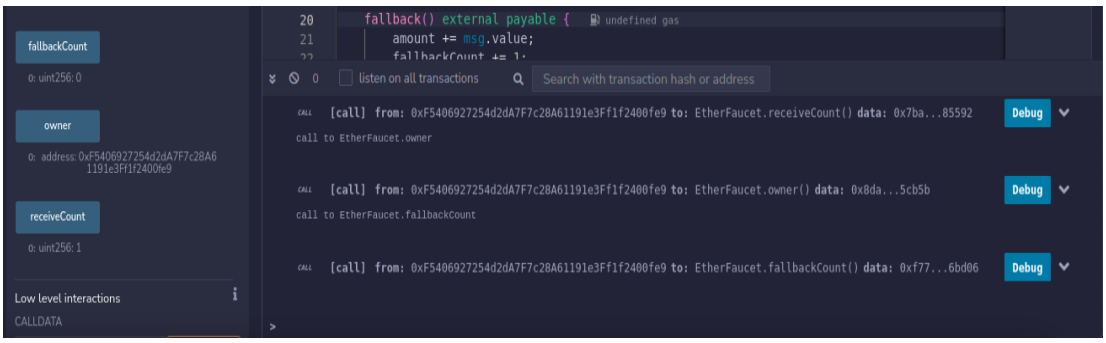


transactionIndex	0
confirmations	3
from	0xF5406927254d2dA7F7c28A61191e3Ff1f2400fe9
gasPrice	1500000007
maxPriorityFeePerGas	1500000000
maxFeePerGas	95791031758
gasLimit	98541
to	0x4bB5df40C3EF59f3Ca28ff416539d8DB35b00F5c
value	5000000000000000000
nonce	17
data	0x
r	0x440af90223a28e35d2b87a08a00dacb61e4d21a6a3dcee8dddd641be69d7fd5d
s	0x28e5ed462271851385 81eea7466405b804c859fd0f50b447d4ed1f9937e75d89
v	0

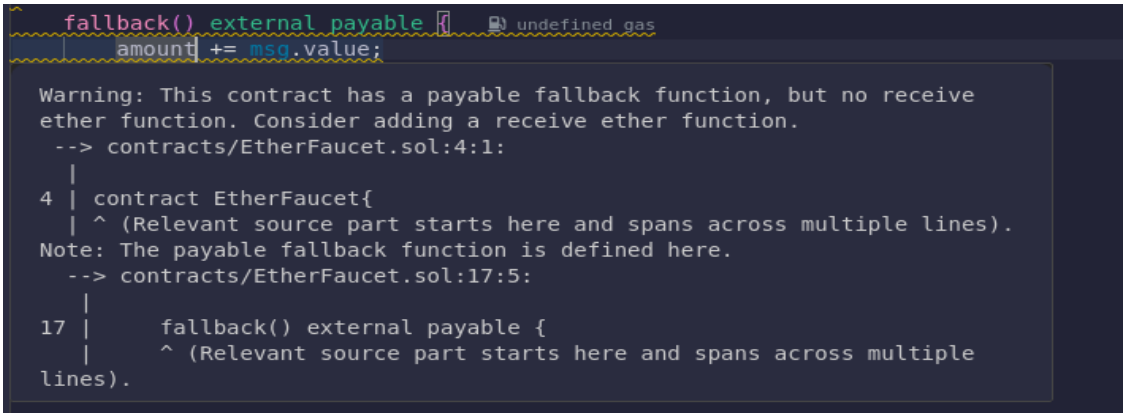
The above screenshot is the block transaction captured when 5 ether was sent from account 1 to EtherFaucel.sol contract.



The above screenshot shows the current balance is 5 Ether received from Metamask.



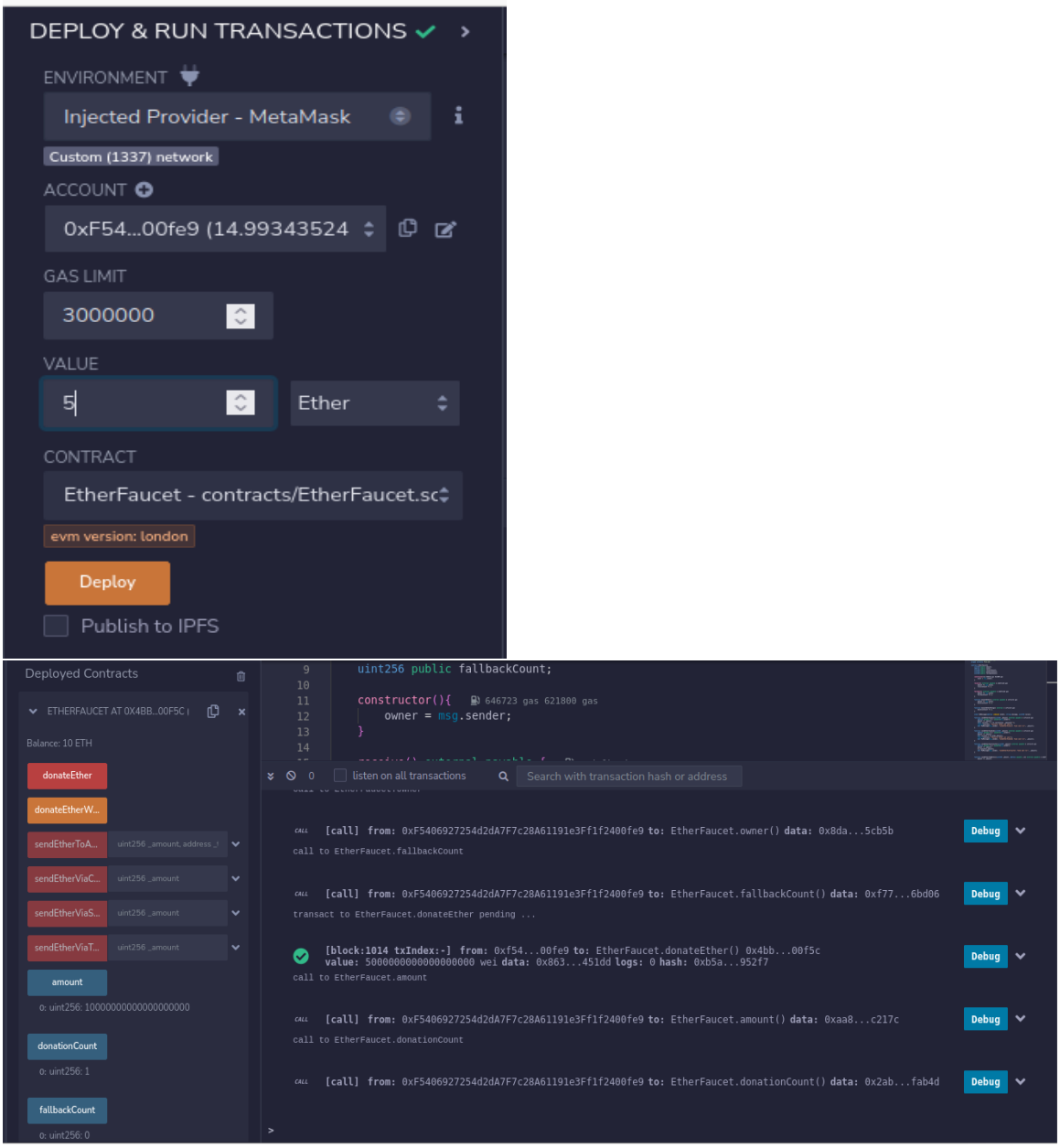
As you can see when money was received, receive() was invoked and receiveCount is 1 whereas fallbackCount is 0.



The above screenshot shows, that when receive() function was removed then a warning is displayed.

Task 3.b: Send fund to a payable function

Here, set the value as 5, which is the value to the donateEther() function.

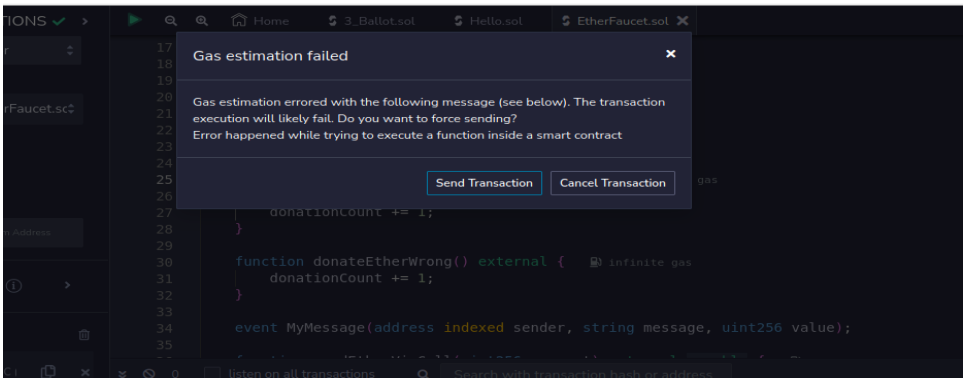


In the above screenshot you can see that the EtherFaucet.sol contract received value 5 and the balance is 10 ether through donateEther(). As you can see Balance is now 10, and amount is also 10 and DonateCount has increased to 1.

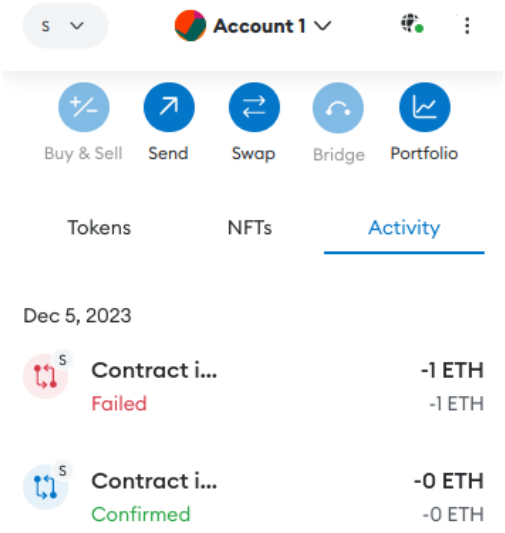
Task 3.c: Send fund to a non-payable function

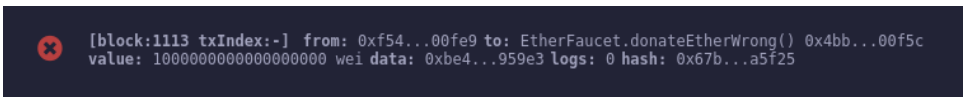
The below warning is thrown when payable keyword is removed from the donateEther() function.

When ether is tried to send to a non payable fucntion donateEtherWrong(), the below error is shown.



The transaction is failed as shown below, the sender will not lose money. Will be charged some gas money for failed transaction.





Task 3.d: Send fund to a non-existing function

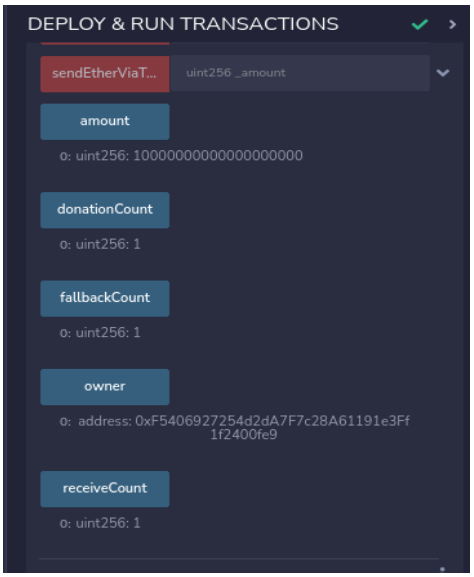
transactionIndex	0
confirmations	2
from	0xF5406927254d2dA7F7c28A61191e3Ff1f2400fe9
gasPrice	1500000007
maxPriorityFeePerGas	1500000000
maxFeePerGas	98062762169
gasLimit	46468
to	0x4bB5df40C3EF59f3Ca28ff416539d8DB35b00F5c
value	0
nonce	21
data	0xc2985578b8f3b75f7dc66a767be2a4ef7d7c2224896a1c86e92ccf30bae678b7
r	0x8b7957a18b52a6b1b2efb69535d8dcd7a1efd11ae7bc4e5dbd75c61f605a78b2
s	0x1855f1c61a2bafcdb0b14bee3f62ad90563d2fb0b20cea3e517d078ea509c65f
v	1

The above block was generated when money was sent to EtherFaucet.sol using foo() function.

```
>>> from web3 import Web3
>>> hash = Web3.sha3(text="foo()")
>>> print(hash.hex())
0xc2985578b8f3b75f7dc66a767be2a4ef7d7c2224896a1c86e92ccf30bae678b7
```

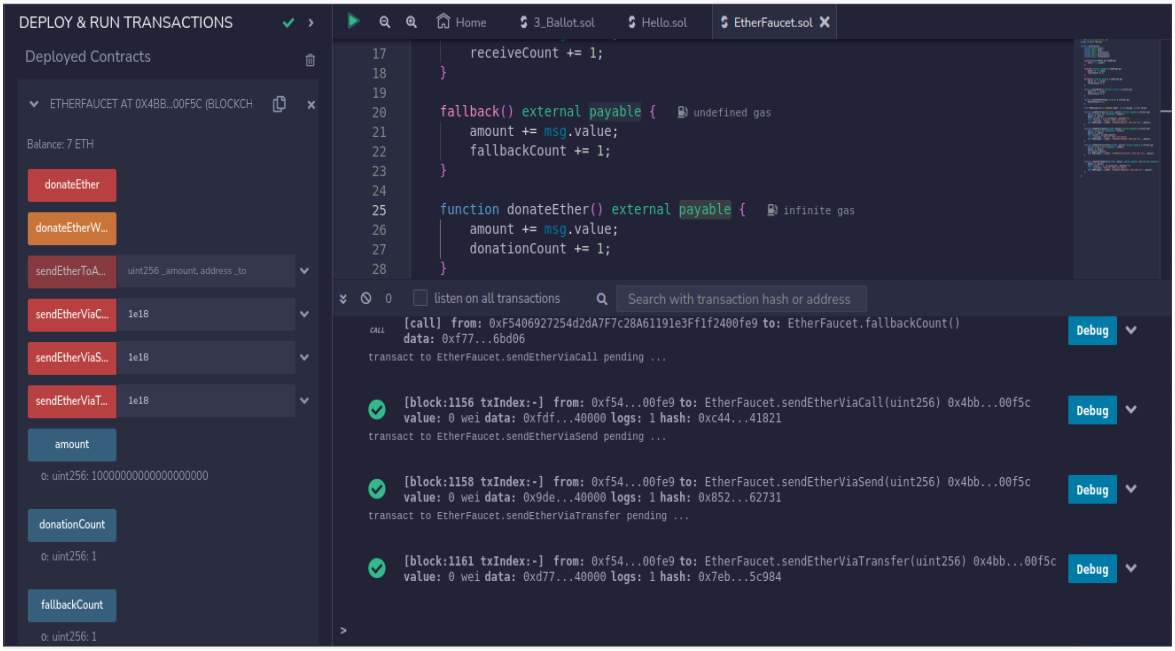
Execute the above python code to get the hex value of foo().

We can see that the successful even though foo() does not exist. When the specified function does not exist fallback() will be invoked. We can see below that fallbackCount() has been increased to 1.



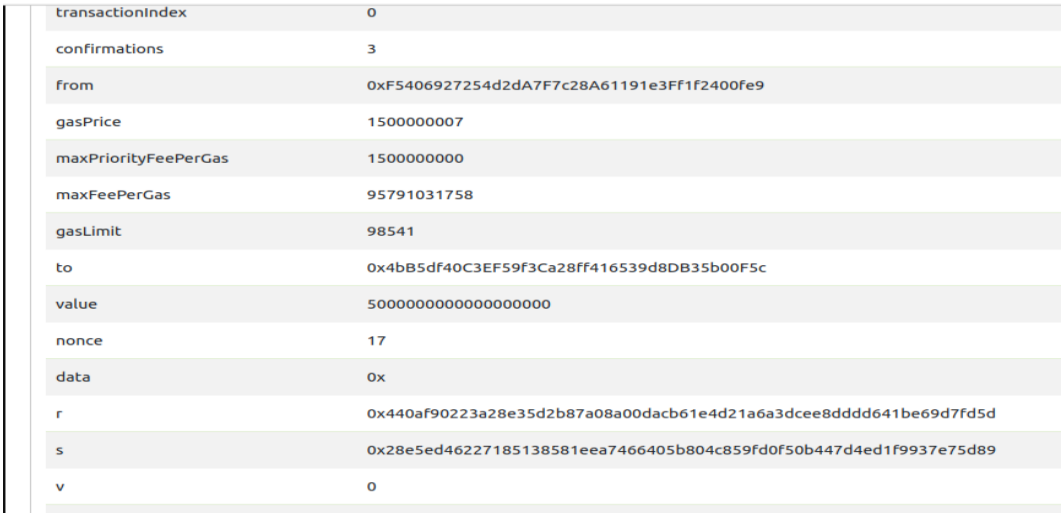
Task 4: Send Fund from Contract

The below screenshot shows that all 3 methods are used to send 1 ether which is 1e1B, and all the 3 transactions are successful and external account received 1 ether for every transaction and the balance from 10 turned to 7.

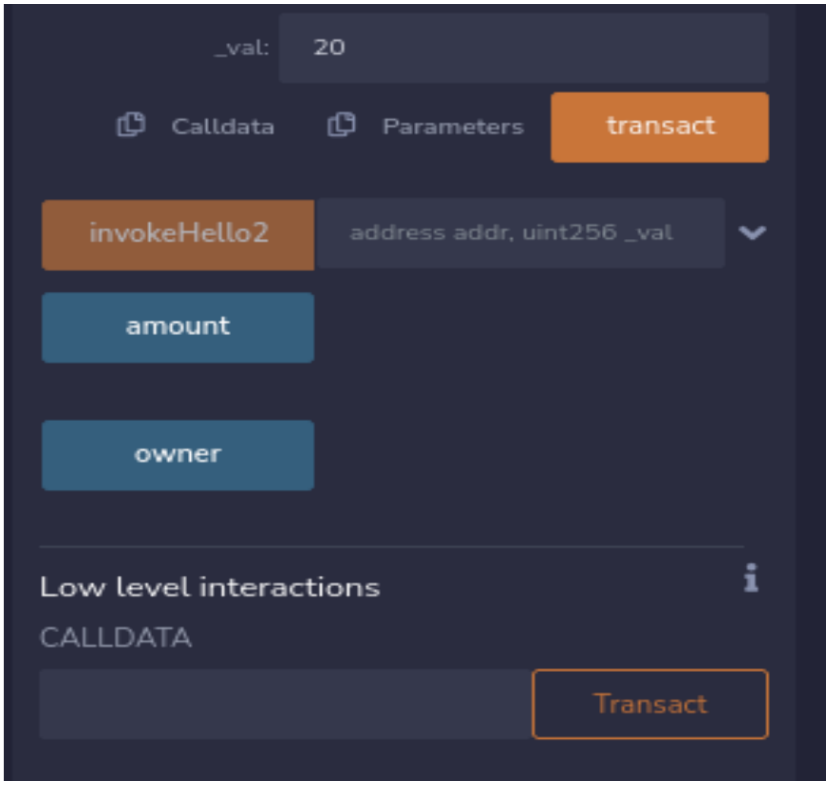


Task 5: Invoke Another Contract

First invoke Hello.sol call and the caller.sol contracts. We deploy the caller contract and use it to invoke two different functions from the Hello contract. The invokeHello() function in the caller.sol contract is a simple invocation, while invokeHello2() sends funds to the Hello contract during the invocation. Hello.sol receives 1 Ether from the caller.sol contract.



transactionIndex	0
confirmations	3
from	0xF5406927254d2dA7F7c28A61191e3Ff1f2400fe9
gasPrice	1500000007
maxPriorityFeePerGas	1500000000
maxFeePerGas	95791031758
gasLimit	98541
to	0x4bB5df40C3EF59f3Ca28ff416539d8DB35b00F5c
value	5000000000000000000
nonce	17
data	0x
r	0x440af90223a28e35d2b87a08a00dacb61e4d21a6a3dcee8ddd641be69d7fd5d
s	0x28e5ed46227185138581eea7466405b804c859fd0f50b447d4ed1f9937e75d89
v	0


_val:
20
Calldata
Parameters
transact
invokeHello2
address addr, uint256 _val
amount
owner
Low level interactions
CALLDATA
Transact


Balance: 0. ETH
increaseCounter
uint256 k
increaseCount...
uint256 k
sendMessage
getCounter
0: uint256: 20
getResult
uint256 a, uint256 b
owner
sayHello
Low level interactions
CALLDATA
Transact