

Dhanashree Srinivasa

SUID: 393473169

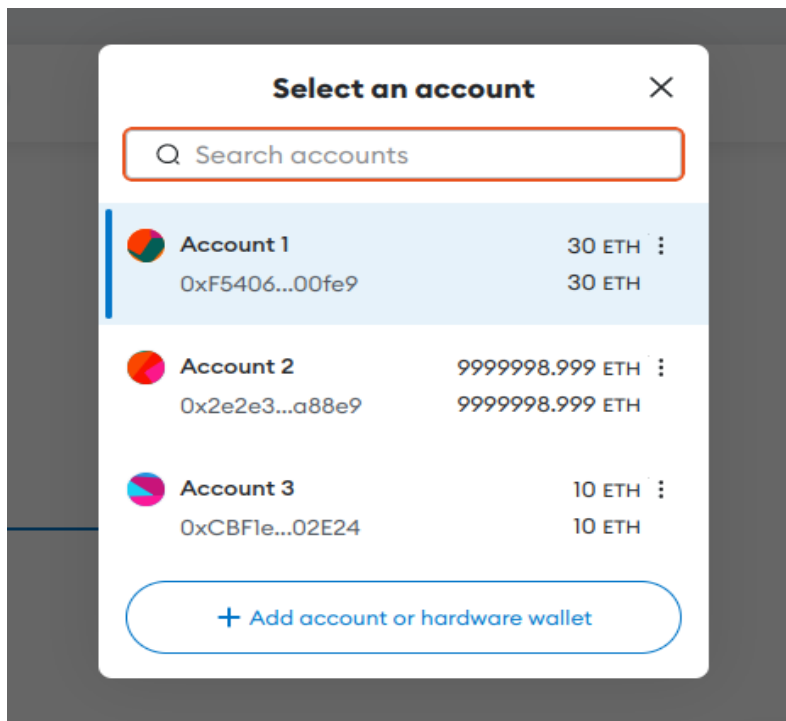
Course: Computer Security - CSE 643

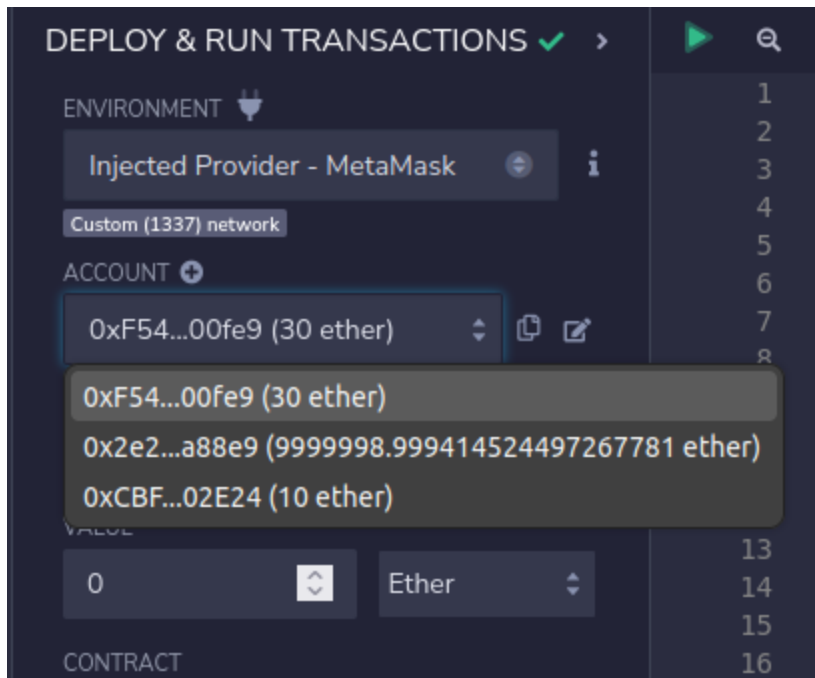
Smart Contract Lab

Task 1: Using Remix for Smart Contract Development

Task 1.a: Connecting Remix to the SEED Emulator

Connect Remix with Metamask SEED Emulator.

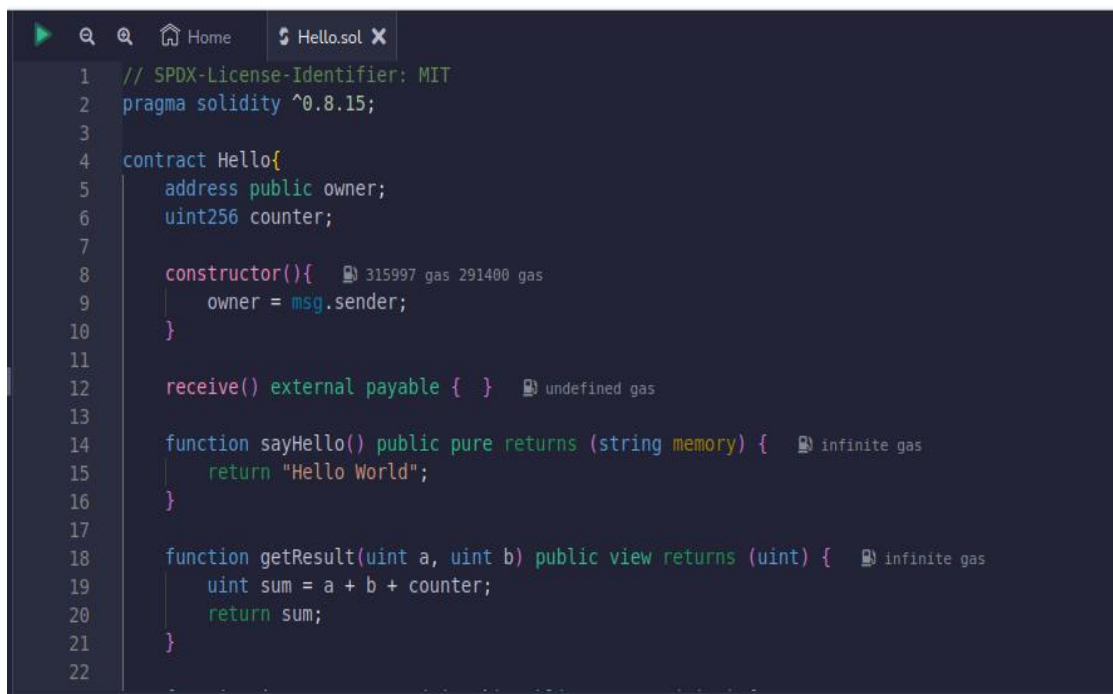
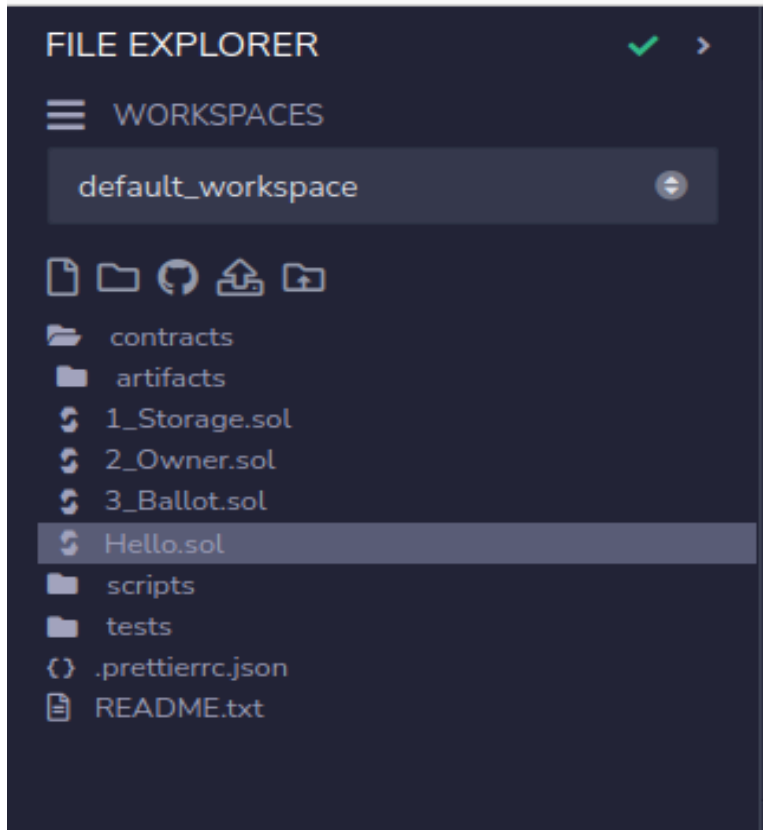




The above screenshot shows Metamask connected to Remix. All 3 accounts are displayed on Remix.

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

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



The screenshot displays the Remix IDE interface, which is used for developing and deploying smart contracts. The interface is divided into two main panels: the **SOLIDITY COMPILER** on the left and the **DEPLOY & RUN TRANSACTIONS** on the right.

SOLIDITY COMPILER Panel:

- COMPILER:** Set to `0.8.15+commit.e14f2714`. Options include ☐ Include nightly builds, ☐ Auto compile, and ☐ Hide warnings.
- Advanced Configurations:** A button to expand more settings.
- Buttons:** `Compile Hello.sol` (blue), `Compile and Run script` (grey), `Publish on Ipfs` (grey), `Publish on Swarm` (grey), and `Compilation Details` (grey).
- CONTRACT:** Set to `Hello (Hello.sol)`.
- Links:** `ABI` and `Bytecode` (both with document icons).

DEPLOY & RUN TRANSACTIONS Panel:

- ENVIRONMENT:** `Injected Provider - MetaMask`. Network: `Custom (1337) network`.
- ACCOUNT:** `0xF54...00fe9 (30 ether)`.
- GAS LIMIT:** `3000000`.
- VALUE:** `0`. Currency: `Ether`.
- CONTRACT:** `Hello - contracts/Hello.sol`. `evm version: london`.
- Buttons:** `Deploy` (orange), `Publish to IPFS` (checkbox), `At Address` (blue), and `Load contract from Address` (grey).
- Transactions recorded:** `1` (with an info icon).

Transaction Log:

```
[block:319 txIndex:-] from: 0x2e2...a88e9 to: Hello.(constructor) value: 0 wei data: 0x608...f0033 logs: 0
hash: 0xbac...1e98a
```

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.

localhost:5000/tx/0xf429a298b5753b20282f1783562d8e540ee14dd13315162d427f7785a5c9bb8b	
Get a new transaction hash	
Transaction	Receipt
hash	0xf429a298b5753b20282f1783562d8e540ee14dd13315162d427f7785a5c9bb8b
type	2
accessList	
blockHash	0xbac7c5ff88580b76b55d74158f9f0efcd7c709166b3e5aab8c12441b9d1e98a
blockNumber	319
transactionIndex	0
confirmations	7
from	0x2e2e3a61daC1A2056d9304F79C168cD16aAa88e9
gasPrice	1500000007
maxPriorityFeePerGas	1500000000
maxFeePerGas	61051245456
gasLimit	390317

Task 2: Invoke Contract Functions

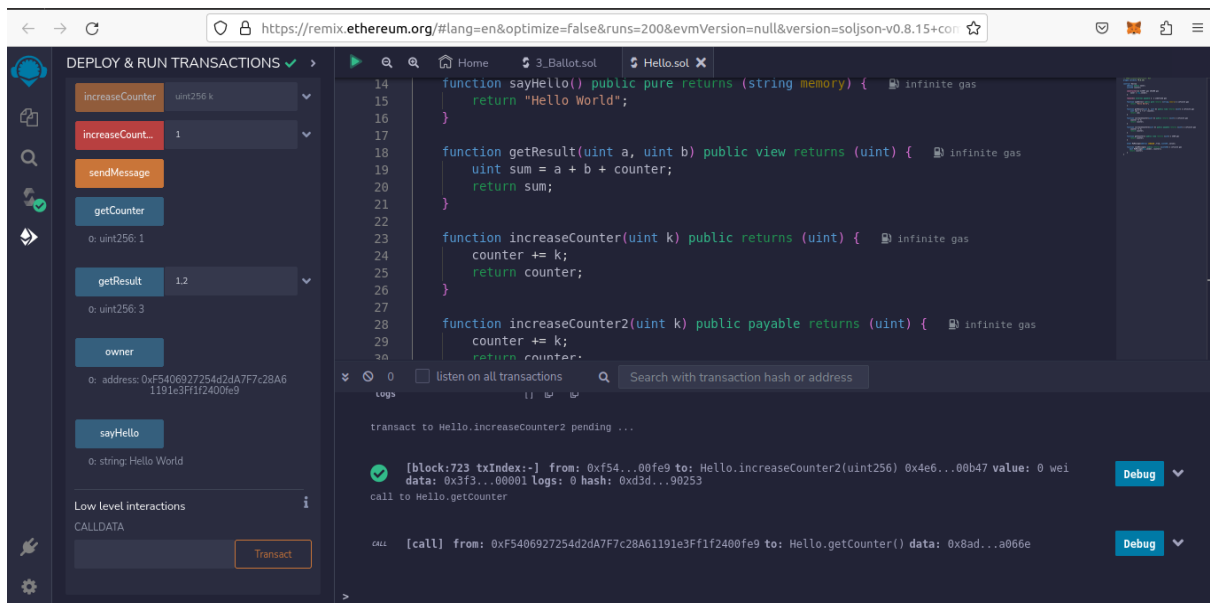
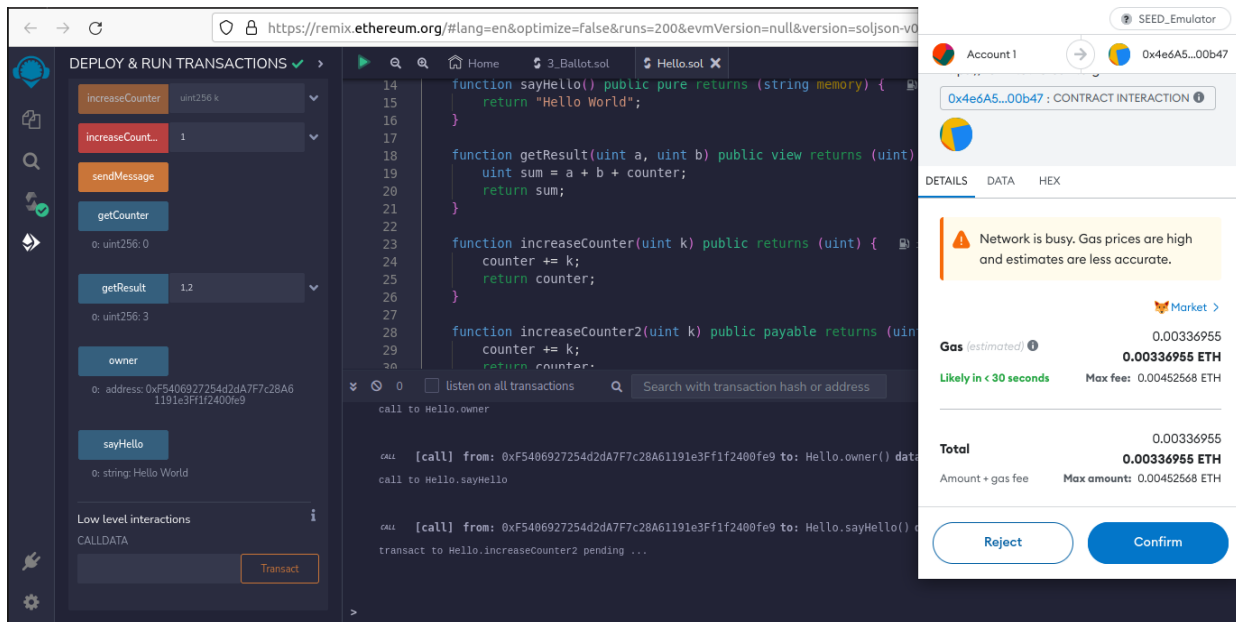
Task 2.a: Invoke a function via local call

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

The screenshot displays the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel shows the contract 'Hello.sol' with functions 'increaseCounter', 'increaseCount...', 'sendMessage', 'getCounter', 'getResult', 'owner', and 'sayHello'. The 'getCounter' function is selected, showing its return value '0x8ad...a066e'. The 'owner' function is also selected, showing its return value '0x8da...5cb5b'. The main editor shows the Solidity code for the 'Hello' contract. The execution log on the right shows several local calls to the contract functions, including 'getCounter', 'getResult', 'owner', and 'sayHello'. The first call to 'getCounter' returns '0x8ad...a066e', and the first call to 'getResult' returns '0x949...00002'. The 'owner' function returns '0x8da...5cb5b', and the 'sayHello' function returns '0xef5...fb05b'.

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.

Get a new transaction hash	
Transaction	Receipt
hash	0x29e3c2074f6156d943f430b31cc06028ac1293094075ebfc1154dd97d8152
type	2
accessList	
blockHash	0xcd7bf4837adb0e722ea367e15d3fa6224e035534d15f5bb24a9fc54b45d9b2c1
blockNumber	700
transactionIndex	0
confirmations	18
from	0xF5406927254d2dA7F7c28A61191e3FF1f2400fe9
gasPrice	1500000007
maxPriorityFeePerGas	1500000000
maxFeePerGas	97441978164
gasLimit	390317

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

```
function decreaseCounter() public returns (uint) {
    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.

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel displays the contract's state: Balance: 0 ETH, and a list of functions including decreaseCounter, increaseCounter, increaseCounter2, sendMessage, getCounter, and getResult. The 'getCounter' function is selected, showing a value of 9. On the right, the 'Hello.sol' contract code is visible, showing the decreaseCounter function. The bottom panel shows the transaction history, including a call to Hello.getCounter() and a call to Hello.decreaseCounter().

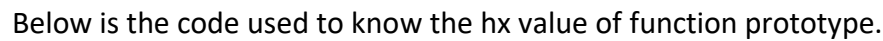
Get a new transaction

hash

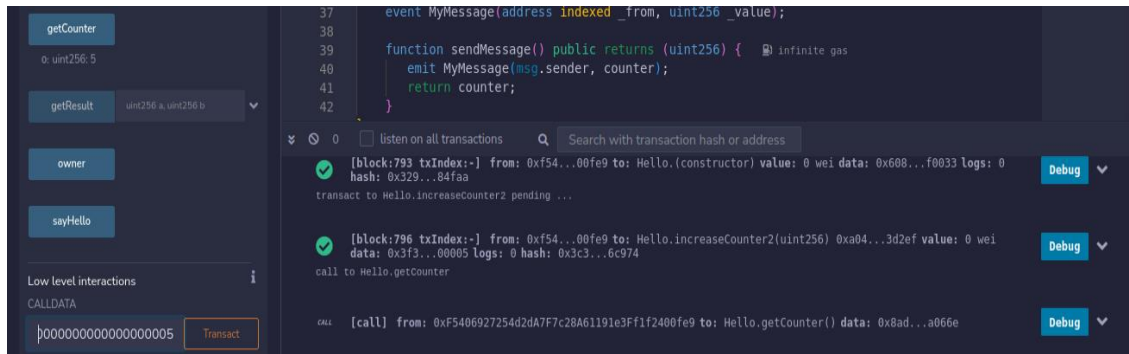
Transaction	Receipt
hash	0x8a40054997e580bc3d48a8faca5d592e01cd153b8cd88270382ff9a864faf111
type	2
accessList	
blockHash	0x82a38d5224158f0d72c95b60725932b813a7e70d21e4296a034413239eacfab
blockNumber	742
transactionIndex	0
confirmations	3
from	0xF5406927254d2dA7F7c28A61191e3Ff1f2400fe9
gasPrice	1500000007
maxPriorityFeePerGas	1500000000
maxFeePerGas	84108233712
gasLimit	26761

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

[illegible]

[illegible]

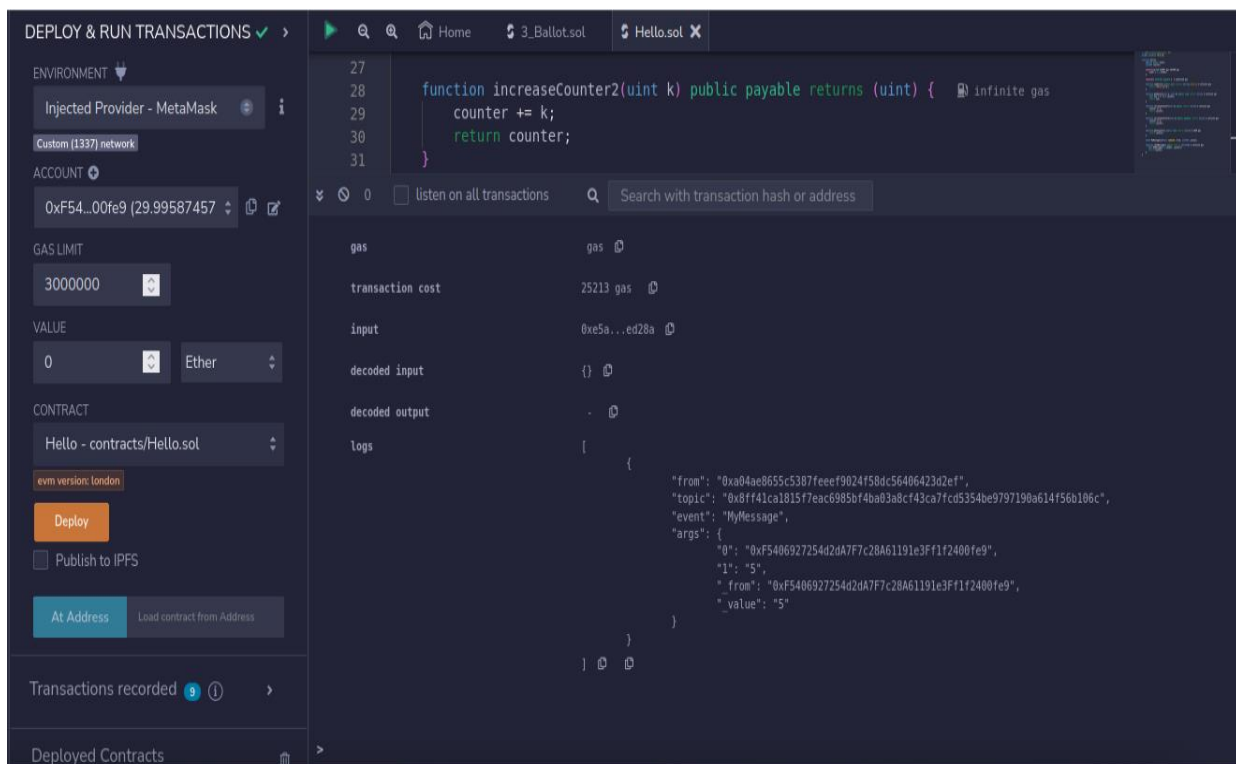
The below screenshot shows that the `increaseCounter()`'s value is increased 5 times.



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

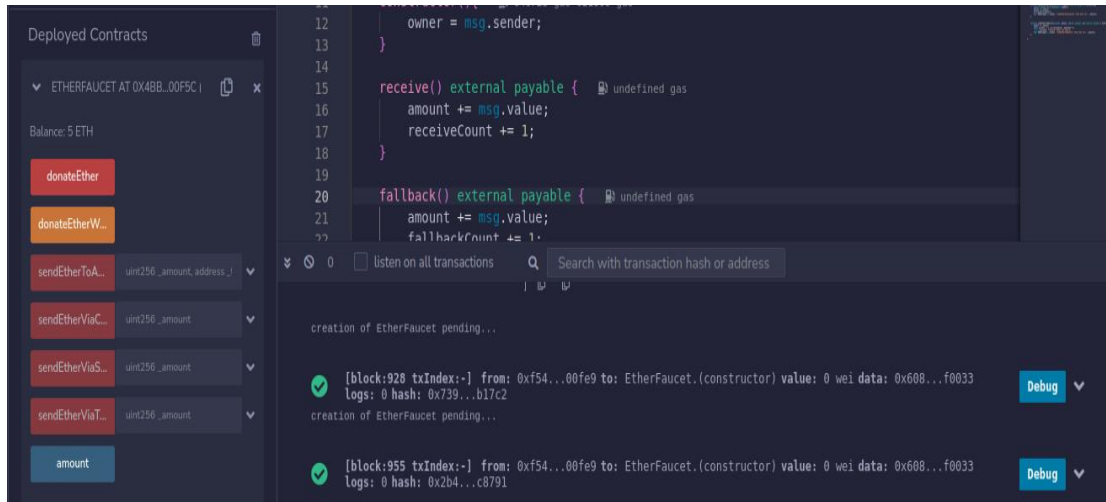
Task 3.a: Send fund directly to a contract address

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

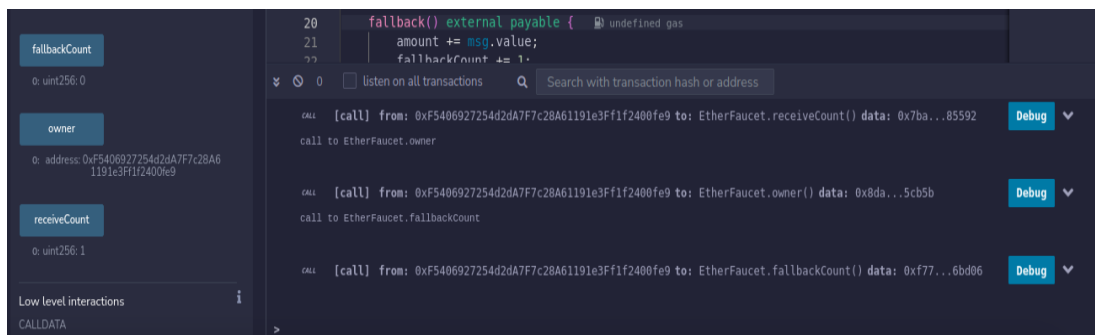
The screenshot shows a 'Send' transaction screen. At the top, there's a header with 'Account 1' and a dropdown arrow. Below the header, the word 'Send' is centered. A text box contains the destination address: '0x4bB5df40C3EF59f3Ca28ff416539d8DB35b00F5c'. Below this, the 'Asset' is set to 'ETH' with a balance of '24.99360698 ETH'. The 'Amount' is set to '5 ETH' with a 'Max' button and a note 'No conversion rate available'. The 'Gas (estimated)' section shows '0.00703043 ETH' and 'Likely in < 30 seconds', with a 'Max fee' of '0.00943934 ETH'. At the bottom, there are 'Cancel' and 'Next' buttons.

transactionIndex	0
confirmations	3
from	0xF5406927254d2dA7F7c28A61191e3FF1f2400fe9
gasPrice	1500000007
maxPriorityFeePerGas	1500000000
maxFeePerGas	95791031758
gasLimit	98541
to	0x4bB5df40C3EF59f3Ca28ff416539d8DB35b00F5c
value	5000000000000000000
nonce	17
data	0x
r	0x440af90223a28e35d2b87a08a00dabc61e4d21a6a3dcee8ddddd641be69d7fd5d
s	0x28e5ed46227185138581eea7466405b804c859fd0f50b447d4ed1f9937e75d89
v	0

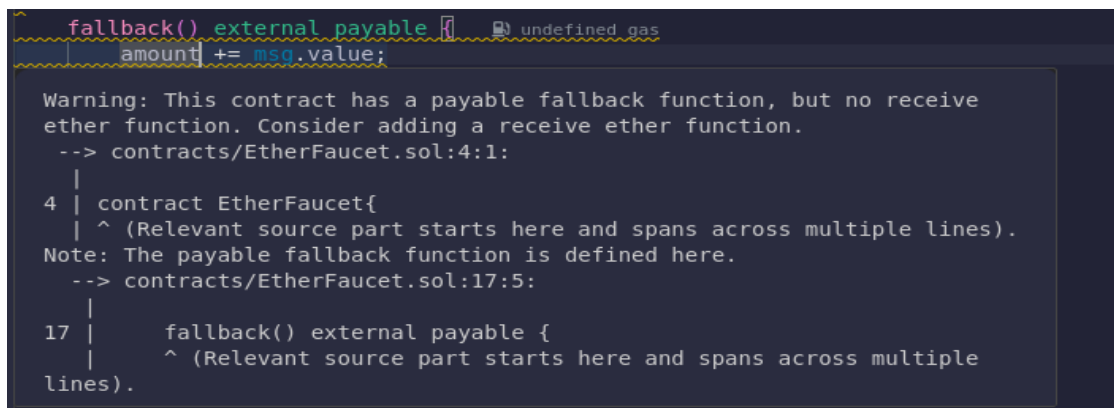
The above screenshot is the block transaction captured when 5 ether was sent from account 1 to EtherFaucet.sol contract.



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.

The screenshot displays the Remix IDE interface, divided into three main sections: the top panel for deployment, the left panel for deployed contracts, and the right panel for the Solidity code and transaction logs.

Top Panel: DEPLOY & RUN TRANSACTIONS

- ENVIRONMENT:** Injected Provider - MetaMask, Custom (1337) network.
- ACCOUNT:** 0xF54...00fe9 (14.99343524 ETH).
- GAS LIMIT:** 3000000.
- VALUE:** 5 Ether.
- CONTRACT:** EtherFaucet - contracts/EtherFaucet.sol, evm version: london.
- Buttons:** Deploy, Publish to IPFS.

Left Panel: Deployed Contracts

- ETHERFAUCET AT 0x4BB...00F5C**
- Balance:** 10 ETH
- Functions:** donateEther, donateEtherW..., sendEtherToA..., sendEtherViaC..., sendEtherViaS..., sendEtherViaT..., amount, donationCount, fallbackCount.

Right Panel: Solidity Code and Transaction Logs

Solidity Code:

```
9  uint256 public fallbackCount;
10
11  constructor(){
12      owner = msg.sender;
13  }
14
```

Transaction Logs:

- CALL** [call] from: 0xF5406927254d2dA7F7c28A61191e3Ff1f2400fe9 to: EtherFaucet.owner() data: 0x8da...5cb5b
- CALL** [call] from: 0xF5406927254d2dA7F7c28A61191e3Ff1f2400fe9 to: EtherFaucet.fallbackCount() data: 0xf77...6bd06
- TRANSACTION** [tx] from: 0xF5406927254d2dA7F7c28A61191e3Ff1f2400fe9 to: EtherFaucet.donateEther() data: 0x4bb...00f5c value: 5000000000000000 wei data: 0x863...451dd logs: 0 hash: 0xb5a...952f7
- CALL** [call] from: 0xF5406927254d2dA7F7c28A61191e3Ff1f2400fe9 to: EtherFaucet.amount() data: 0xaa8...c217c
- CALL** [call] from: 0xF5406927254d2dA7F7c28A61191e3Ff1f2400fe9 to: EtherFaucet.donationCount() data: 0x2ab...fab4d

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

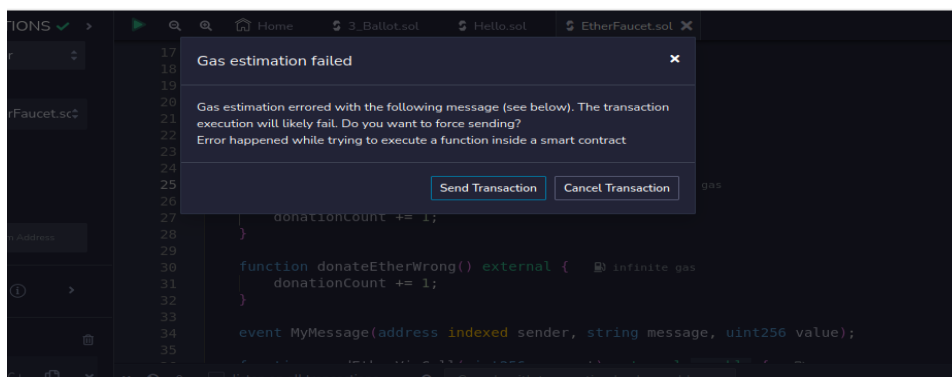
The below warning is thrown when payable keyword is removed from the donateEther() function.

```

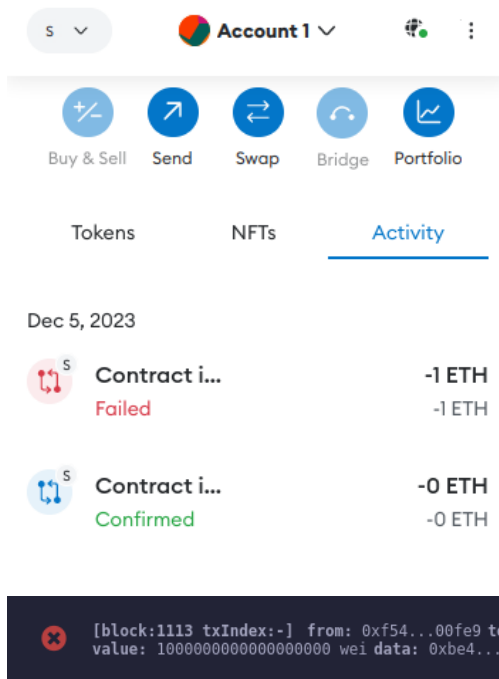
21 |         amount += msg.value;
22 |         fallbackCount += 1;
23 |
24 | from solidity:
25 | TypeError: "msg.value" and "callvalue()" can only be used in payable public functions. Make the function "payable" or use an internal function to
    | => contracts/EtherFaucet.sol:26:19:
26 |         amount += msg.value;
    |
31 |         donationCount += 1;

```

When ether is tried to send to a non payable function 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.



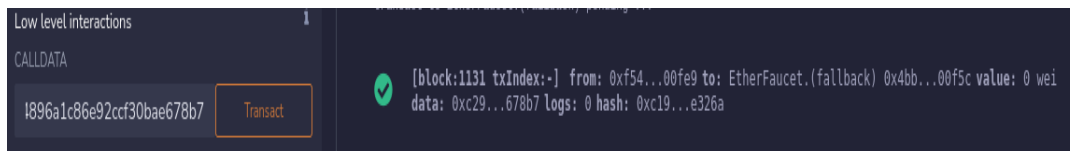
Task 3.d: Send fund to a non-existing function

transactionIndex	0
confirmations	2
from	0xF5406927254d2dA7F7c28A61191e3FF1f2400Fe9
gasPrice	1500000007
maxPriorityFeePerGas	1500000000
maxFeePerGas	98062762169
gasLimit	46468
to	0x4bB5df40C3EF59f3Ca28ff416539d8DB35b00F5c
value	0
nonce	21
data	0xcc2985578b8f3b75f7dc66a767be2a4ef7d7c2224896a1c86e92ccf30bae678b7
r	0x8b7957a18b52a6b1b2efb69535d8dcd7a1efd11ae7bc4e5dbd75c61f605a78b2
s	0x1855f1c61a2bafcdb0b14bee3f62ad90563d2fb0b20cea3e517d078ea509c65f
v	1

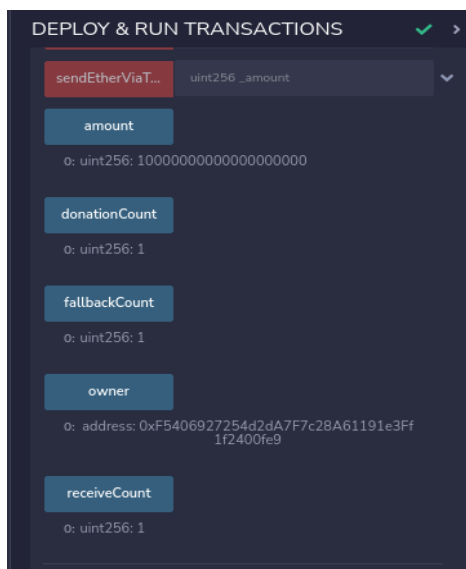
The above block was generated when money was sent to EtherFaucet.sol using foo() function.

```
>>> 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.

The screenshot displays the 'DEPLOY & RUN TRANSACTIONS' interface. On the left, the 'Deployed Contracts' section shows 'ETHERFAUCET AT 0x4BB...00F5C (BLOCKCH)'. The main area shows the Solidity code for 'EtherFaucet.sol', which includes a fallback function and a donateEther function. The bottom panel shows the transaction history, listing three successful transactions: a call to fallbackCount(), a sendEtherViaCall(), and a sendEtherViaTransfer().

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	0x440af90223a28e35d2b87a08a00dadb61e4d21a6a3dcee8ddddd641be69d7fd5d
s	0x28e5ed46227185138581eea7466405b804c859fd0f50b447d4ed1f9937e75d89
v	0

Low level interactions

CALLDATA

Balance: 0. ETH

o: uint256: 20

Low level interactions

CALLDATA