

Riley Thompson
826526487

Blockchain Lab 2

1.

Called the greeter function with parameter hey!, and then when I called greet, it displays hey!

The screenshot shows the Truffle UI interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is open, showing a transaction record for deploying the 'hello' contract from address 0x5B3...eddC4. The transaction details include a value of 0 Wei and an evm version of byzantium. The 'Transactions recorded' section shows the deployment of the 'hello' contract at address 0x5B8E...9FB8. The 'Deployed/Unpinned Contracts' section lists the deployed 'hello' contract with its address and balance of 0 ETH. Under 'Low level interactions', there are two entries: 'greeter' (with value Hey!) and 'greet' (with value string: Hey!). On the right, the code editor shows the Solidity source code for GreetingsA2.sol:

```
pragma solidity ^0.4.26;
contract hello {
    /* define variable greeting of the type string */
    string greeting;
    function greeter(string _greeting) public {
        greeting = _greeting;
    }
    function greet() public constant returns(string) {
        return greeting;
    }
}
```

A modal window titled 'Riley Thompson' is displayed over the code editor, containing the text 'Riley Thompson'.

2.

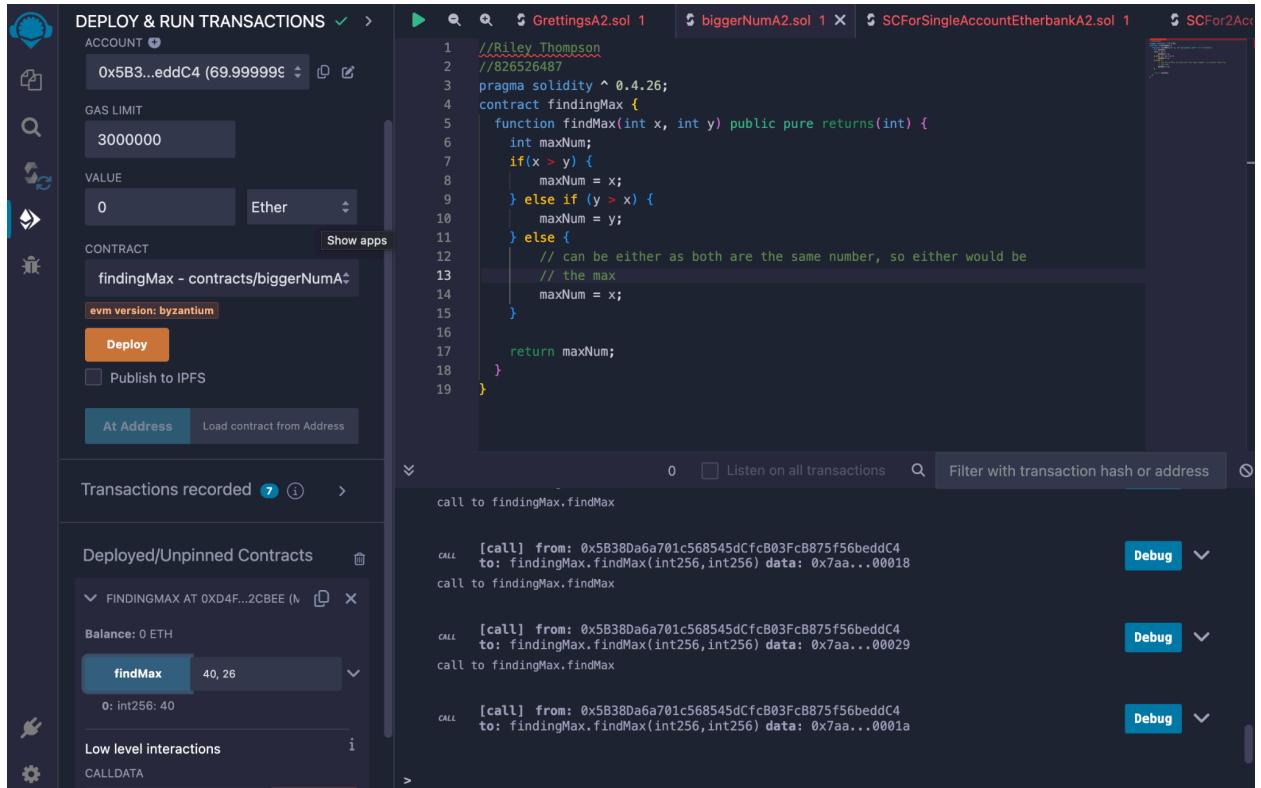
Shows when y is greater than x, returns y

The screenshot shows the Truffle UI interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is open, showing a transaction record for deploying the 'findingMax' contract from address 0x5B3...eddC4. The transaction details include a value of 0 Ether and an evm version of byzantium. The 'Transactions recorded' section shows the deployment of the 'findingMax' contract at address 0xD4F...2CBEE. The 'Deployed/Unpinned Contracts' section lists the deployed 'findingMax' contract with its address and balance of 0 ETH. Under 'Low level interactions', there is one entry: 'findMax' with values 25, 41. On the right, the code editor shows the Solidity source code for biggerNumA2.sol:

```
//@Riley_Thompson
//826526487
pragma solidity ^ 0.4.26;
contract findingMax {
    function findMax(int x, int y) public pure returns(int) {
        int maxNum;
        if(x > y) {
            maxNum = x;
        } else if (y > x) {
            maxNum = y;
        } else {
            // can be either as both are the same number, so either would be
            // the max
            maxNum = x;
        }
        return maxNum;
    }
}
```

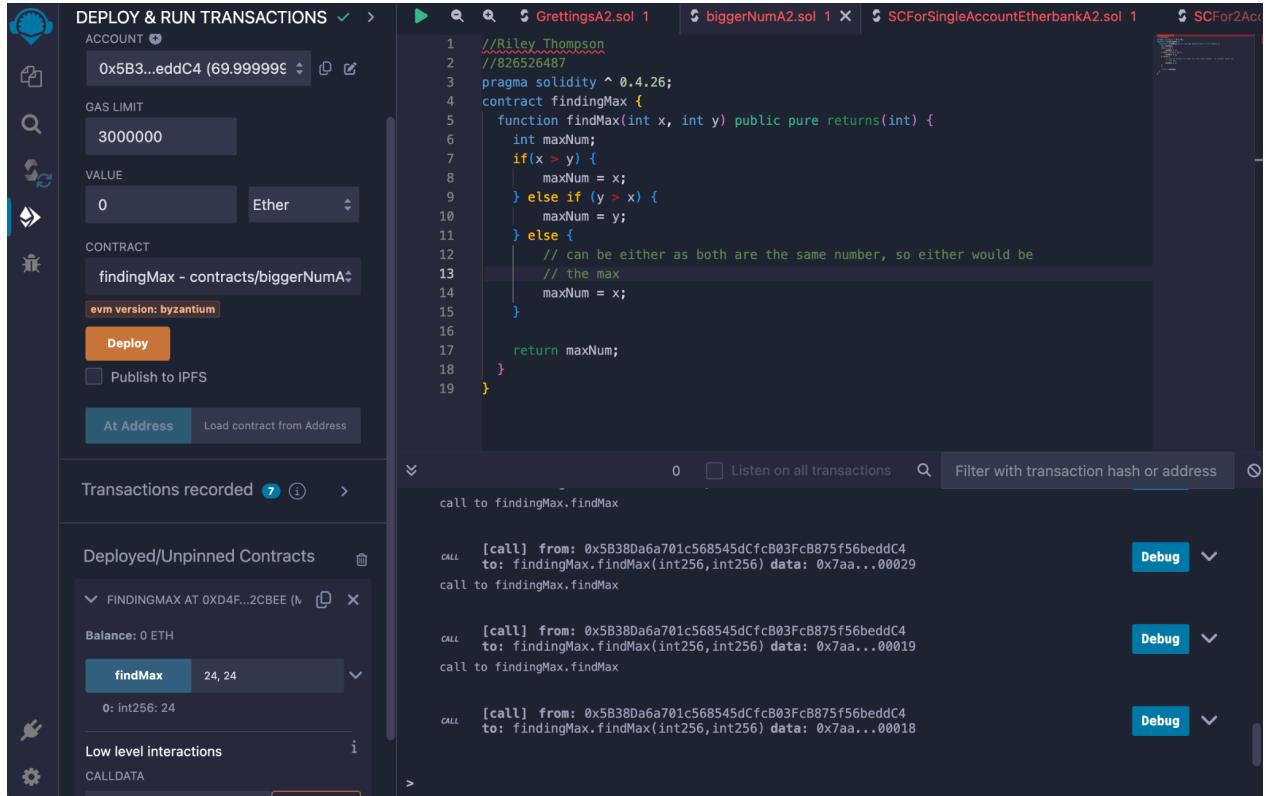
The transaction history on the right shows three calls to the 'findMax' function: one with arguments 25, 41, and two more recent ones with arguments 256, 256.

Shows when x is greater than y, returns x



```
//Riley Thompson
//026526487
pragma solidity ^ 0.4.26;
contract findingMax {
    function findMax(int x, int y) public pure returns(int) {
        int maxNum;
        if(x > y) {
            maxNum = x;
        } else if (y > x) {
            maxNum = y;
        } else {
            // can be either as both are the same number, so either would be
            // the max
            maxNum = x;
        }
        return maxNum;
    }
}
```

Shows when they are equal and would return x as they are the same, so could be x or y



```
//Riley Thompson
//026526487
pragma solidity ^ 0.4.26;
contract findingMax {
    function findMax(int x, int y) public pure returns(int) {
        int maxNum;
        if(x > y) {
            maxNum = x;
        } else if (y > x) {
            maxNum = y;
        } else {
            // can be either as both are the same number, so either would be
            // the max
            maxNum = x;
        }
        return maxNum;
    }
}
```

The file that is submitted is biggerNumA2.sol

3.

After deployment, click the 'getBalanceCA' button to show the balance(which is 0):

The screenshot shows the Truffle UI interface. On the left, there's a sidebar with icons for network selection, deployment status, and publishing to IPFS. The main area has tabs for 'DEPLOY & RUN TRANSACTIONS' and 'CONTRACTS'. Under 'DEPLOY & RUN TRANSACTIONS', it shows a deployed contract named 'bank - contracts/SCForSingleAccountEtherBank' with EVM version 'byzantium'. It has a 'Deploy' button and a 'Publish to IPFS' option. Below that, it lists 'Transactions recorded' (9) and 'Deployed/Unpinned Contracts' (1), which is 'BANK AT 0X93F...C96CC (MEMORY)' with a balance of 0 ETH. There are buttons for 'deposit', 'relay', 'withdraw', 'getBalanceCA', and 'getBalanceEOA'. The 'getBalanceCA' button is highlighted. On the right, the code editor shows the Solidity source code for 'rNumA2.sol' (version 0.4.26). The code defines a 'bank' contract with functions for depositing ether, withdrawing ether, relaying ether to another address, and two methods to get the balance of an account ('getBalanceCA' and 'getBalanceEOA'). The 'getBalanceCA' method returns the balance in wei. The code editor also shows some transaction logs from the blockchain.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.4.26;
contract bank {
    uint256 EtherBalance_Alice = 0;
    function deposit() public payable {
        EtherBalance_Alice = EtherBalance_Alice + msg.value;
    }
    function withdraw(uint256 ethers) public payable {
        msg.sender.transfer(ethers * 1000000000000000000);
        EtherBalance_Alice = EtherBalance_Alice - ethers;
    }
    function relay(address Bob) public payable {
        Bob.transfer(msg.value);
    }
    function getBalanceCA() public constant returns(uint256){
        return EtherBalance_Alice;
    }
    function getBalanceEOA() public view returns(uint256) {
        return 0x00000000000000000000000000000000.balance;
    }
}
```

Deposited from Alice Adress with Ether 10, so displays 10000000000000000000 wei which is 10 ether:

The screenshot shows the Truffle UI interface with several tabs open:

- DEPLOY & RUN TRANSACTIONS**:
 - Value: 0 Ether
 - Contract: bank - contracts/SCForSingleAccountA2.sol
 - EVM version: byzantium
 - Deploy button
 - Publish to IPFS checkbox
 - At Address and Load contract from Address buttons
- Transactions recorded**: Shows 10 transactions.
- Deployed/Unpinned Contracts**: Shows a deployed contract BANK AT 0x93f...C96CC (MEMORY).
- Low level interactions**: Shows methods for the deployed contract:
 - deposit
 - relay address Bob
 - withdraw uint256 ethers
 - getBalanceCA
 - getBalanceEOA

The right side of the screen displays the Solidity code for the contract rNumA2.sol 1:

```
1 //Riley Thompson
2 //826526487
3 pragma solidity ^ 0.4.26;
4 contract bank {
5     uint256 EtherBalance_Alice = 0;
6     function deposit() public payable {
7         EtherBalance_Alice = EtherBalance_Alice + msg.value;
8     }
9     function withdraw(uint256 ethers) public payable {
10        msg.sender.transfer(ethers * 1000000000000000000);
11        EtherBalance_Alice = EtherBalance_Alice - ethers;
12    }
13    function relay(address Bob) public payable {
14        Bob.transfer(msg.value);
15    }
16    function getBalanceCA() public constant returns(uint256){
17        return EtherBalance_Alice;
18    }
19    function getBalanceEOA() public view returns(uint256) {
20        return 0x00000000000000000000000000000000.balance;
21    }
}
```

The transaction history pane shows:

- A call to bank.getBalanceCA.
- A transaction to bank.deposit pending ...
- A successful transaction [vm] from: 0x5B380a6a701c568545dCfcB03FcB875f56beddC4 to: bank.getBalanceCA(). Value: 100000000000000000000000000000000 Wei data: 0xd0e...30db0 logs: 0 hash: 0x896...028fd
- A call to bank.getBalanceCA.
- A transaction to bank.getBalanceCA pending ...

Buttons for Debugging are available for each transaction.

Called relay function with argument

0x00000000000000000000000000000000 and getBalanceEOA, returns the balance of that address which is at 20 ether or 20000000000000000000 wei

The screenshot shows the Truffle UI interface with the following details:

- Deploy & Run Transactions**:
 - Bank = contracts/SCForSingleAccount
 - evm version: byzantium
 - Deploy** button
 - Publish to IPFS
 - At Address: Load contract from Address
- Transactions recorded**: 19
- Deployed/Unpinned Contracts**: BANK AT 0xE5f...78e22 (MEMORY)
 - Balance: 10 ETH
 - deposit** button
 - relay**: 0x00000000000000000000000000000000
 - withdraw**: uint256 ethers
 - getBalanceCA**: 0: uint256: 10000000000000000000000000000000
 - getBalanceEOA**: 0: uint256: 20000000000000000000000000000000
- Low level interactions**: CALLDATA
- Transact** button
- rNum2A.sol 1**:
 - pragma solidity ^ 0.4.26;
 - contract bank {
 - uint256 EtherBalance_Alice = 0;
 - function deposit() public payable {
 - EtherBalance_Alice = EtherBalance_Alice + msg.value;
 - }
 - function withdraw(uint256 ethers) public payable {
 - msg.sender.transfer(ethers * 10000000000000000000000000000000);
 - EtherBalance_Alice = EtherBalance_Alice - ethers;
 - }
 - function relay(address Bob) public payable {
 - Bob.transfer(msg.value);
 - }
 - function getBalanceCA() public constant returns(uint256){
 - return EtherBalance_Alice;
 - }
 - function getBalanceEOA() public view returns(uint256) {
 - return 0x00000000000000000000000000000000.balance;
 - }
 - 0 Listen on all transactions
 - Filter with transaction hash or address
- call to bank.getBalanceCA**
 - [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: bank.getBalanceCA()
 - data: 0x2f9...febd3
 - transact to bank.relay pending ...
- call to bank.getBalanceEOA**
 - [vm] from: 0x5B3...eddC4 to: bank.relay(address) 0xE5f...78e22 value: 0 wei
 - data: 0x73b...00000 logs: 0 hash: 0xea2...79ac1
 - call to bank.getBalanceEOA()
- call to bank.getBalanceEOA**
 - [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: bank.getBalanceEOA()
 - data: 0xa84...c1d07

4.

I put public at the withdraw function as it gave warning otherwise and didn't like having it have warning and said just defaults to public anyways, so I put public on withdraw function:

```
contracts/SCFor2AccountEtherBan
k.sol:26:5: Warning: No
visibility specified.
Defaulting to "public".
function withdraw(uint amount)
{
^ (Relevant source part starts
here and spans across multiple
lines).
```

I began with depositing 5 and 3 from Alice(5 ether), then Bob(3 ether):

The screenshot shows the Truffle UI interface. On the left, there's a sidebar with various icons. The main area has tabs for 'DEPLOY & RUN TRANSACTIONS' and 'Deployed/Unpinned Contracts'. Under 'DEPLOY & RUN TRANSACTIONS', there's a 'CONTRACT' section where 'bank_m' is selected, and a 'Deploy' button with 'address Alice, address Bob' selected. Below that are buttons for 'Save' and 'Run'. Under 'Transactions recorded', there's a checkbox for 'Run transactions using the latest compilation result' and buttons for 'Save' and 'Run'. In the center, the code for 'bank_m' is displayed in a code editor:

```
//Riley.Thompson
//826526487
pragma solidity ^0.4.26;
contract bank_m {
    address private AliceA;
    address private BobA;

    uint private AliceBalance;
    uint private BobBalance;
    uint private totalBalance;

    constructor(address Alice, address Bob) public {
        AliceA = Alice;
        BobA = Bob;
    }

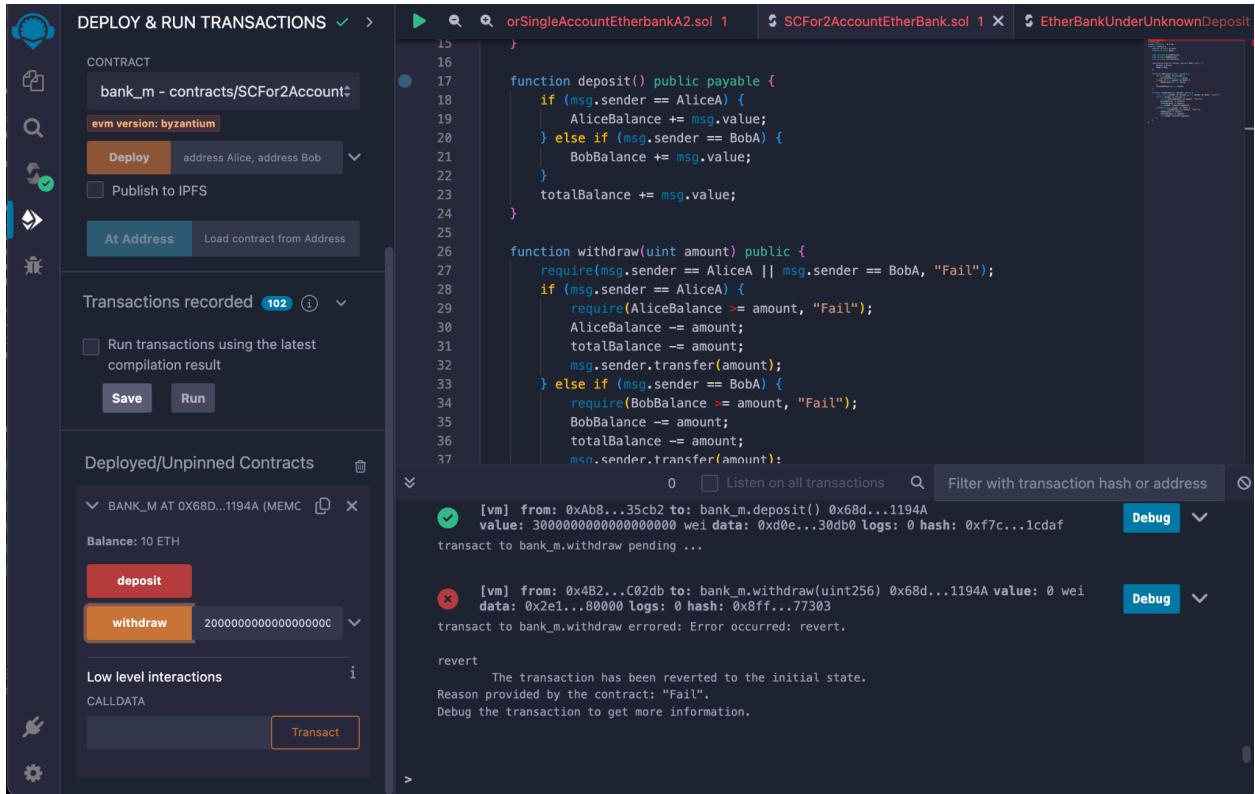
    function deposit() public payable {
        if (msg.sender == AliceA) {
            AliceBalance += msg.value;
        } else if (msg.sender == BobA) {
            BobBalance += msg.value;
        }
    }
}
```

On the right, there's a transaction history pane. It shows two transactions from the 'BANK_M' contract:

- Transaction 1: [vm] from: 0x5B3...eddC4 to: bank_m.deposit() 0x68d...1194A value: 500000000000000000 wei data: 0xd0e...30db0 logs: 0 hash: 0xf7...f7246
- Transaction 2: [vm] from: 0xAB8...35cb2 to: bank_m.deposit() 0x68d...1194A value: 300000000000000000 wei data: 0xd0e...30db0 logs: 0 hash: 0x91b...4f8ab

Each transaction has a 'Debug' button next to it.

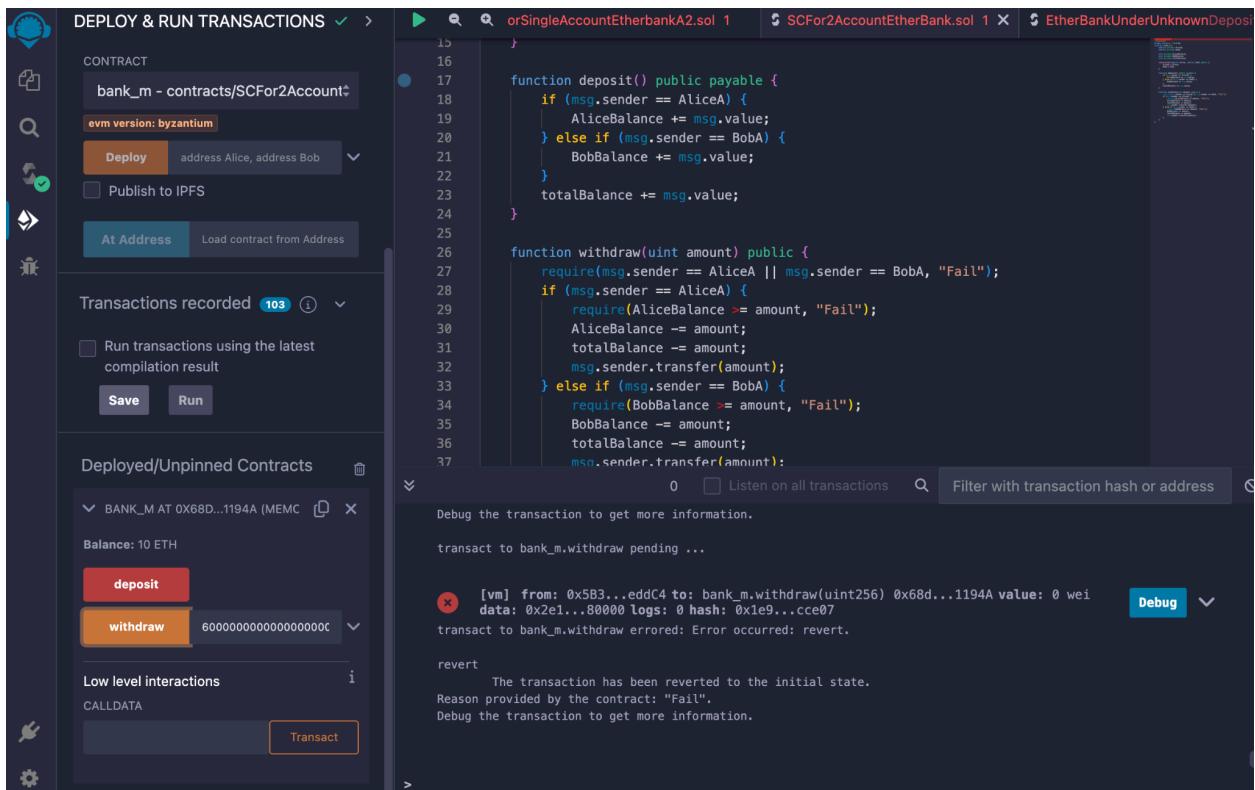
Charlie's address fails to get money from bank(correct) - only Alice and Bob should:



The screenshot shows the Truffle UI interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is open, showing a deployed contract named 'BANK_M' with a balance of 10 ETH. In the center, the code editor displays the Solidity smart contract 'orSingleAccountEtherbankA2.sol'. The 'withdraw' function is highlighted. On the right, the transaction history shows a successful withdrawal from Alice's account:

```
[vm] from: 0xAb8...35cb2 to: bank_m.deposit() 0x68d...1194A
value: 3000000000000000000 wei data: 0xd0e...30db0 logs: 0 hash: 0xf7c...1cdf
transact to bank_m.withdraw pending ...
```

Cannot withdraw more than amount in own address amount(success) - Alice trying to withdraw 6 ether:



The screenshot shows the Truffle UI interface. The setup is identical to the previous one, with a deployed 'BANK_M' contract and 10 ETH balance. The transaction history on the right shows a failed withdrawal attempt by Alice:

```
[vm] from: 0x4B2...C02db to: bank_m.withdraw(uint256) 0x68d...1194A value: 0 wei
data: 0x2e1...80000 logs: 0 hash: 0x8ff...77303
transact to bank_m.withdraw errored: Error occurred: revert.
```

A 'revert' message follows, indicating the transaction was reverted due to the 'Fail' condition in the contract's logic.

Able to withdraw what own address amount has(success):

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is open, showing a deployed contract named 'BANK_M' at address 0x68d...1194A. It has a balance of 6 ETH. There are buttons for 'deposit' and 'withdraw'. The 'withdraw' button is highlighted with the value '4000000000000000000'. Below it, under 'Low level interactions', there is a 'CALLDATA' section with a 'Transact' button. On the right, the code editor displays two Solidity files: 'orSingleAccountEtherbankA2.sol' and 'SCFor2AccountEtherBank.sol'. The code for 'orSingleAccountEtherbankA2.sol' includes a withdrawal function that checks if the sender is Alice or Bob and then transfers the specified amount from their respective balances to the total balance. A transaction history panel shows two transactions: one revert and one successful withdrawal of 0 wei.

```
function withdraw(uint amount) public {
    require(msg.sender == AliceA || msg.sender == BobA, "Fail");
    if (msg.sender == AliceA) {
        AliceBalance -= amount;
        totalBalance -= amount;
        msg.sender.transfer(amount);
    } else if (msg.sender == BobA) {
        require(BobBalance >= amount, "Fail");
        BobBalance -= amount;
        totalBalance -= amount;
        msg.sender.transfer(amount);
    }
}
```

[vm] from: 0x5B3...eddC4 to: bank_m.withdraw(uint256) 0x68d...1194A value: 0 wei
data: 0x2e1...00000 logs: 0 hash: 0x6da...876a4 Debug

Other test cases work, tested(success):

Here is my code in a picture:

This screenshot shows the Remix IDE with a different configuration. The environment is set to 'Remix VM (Cancun)' and the account is set to 0x5B3...eddC4 with a balance of 43.9999999. The gas limit is set to 600000. The code editor shows the same Solidity files as the previous screenshot. The transaction history panel shows a successful withdrawal of 0 wei from the contract to the account 0x2e1...00000.

```
function withdraw(uint amount) public {
    require(msg.sender == AliceA || msg.sender == BobA, "Fail");
    if (msg.sender == AliceA) {
        AliceBalance -= amount;
        totalBalance -= amount;
        msg.sender.transfer(amount);
    } else if (msg.sender == BobA) {
        require(BobBalance >= amount, "Fail");
        BobBalance -= amount;
        totalBalance -= amount;
        msg.sender.transfer(amount);
    }
}
```

[vm] from: 0x5B3...eddC4 to: bank_m.withdraw(uint256) 0x68d...1194A value: 0 wei
data: 0x2e1...00000 logs: 0 hash: 0x6da...876a4 Debug

The file that is submitted is SCFor2AccountEtherBank.sol

5.

I put public at the withdraw function as it gave warning otherwise and didn't like having it have warning and said just defaults to public anyways, so I put public on withdraw function:

```
contracts/SCFor2AccountEtherBan
k.sol:26:5: Warning: No
visibility specified.
Defaulting to "public".
function withdraw(uint amount)
{
^ (Relevant source part starts
here and spans across multiple
lines).
```

1) Charlie sends a basic Ethereum transaction that calls no function in bank_m.

-Added:

```
// for new solidity, fallback() didn't work and compiler failed,
// but this is the fallback function and does the same thing
// changed fallback() to function()
// ensures that Charlie can't send a basic Ethereum transaction that calls no function in bank_m.
function() external payable {
    require(msg.sender == AliceA || msg.sender == BobA, "Fail");
    if (msg.sender == AliceA) {
        AliceBalance += msg.value;
    } else if (msg.sender == BobA) {
        BobBalance += msg.value;
    }
}
```

2) Charlie sends a transaction to externally call the deposit() function in bank_m explicitly.

-Added this in deposit() function:

```
// Only Alice and Bob are able to use, otherwise fails
require(msg.sender == AliceA || msg.sender == BobA, "Fail");
```

Here is testing the code. Nothing changed aside from adding things that restrict Charlie address from sending Ether, so deposit and withdrawal work as above in problem 4, and now going to show that cannot deposit with Allice or Bob:

The screenshot shows the Truffle UI interface. On the left, the "DEPLOY & RUN TRANSACTIONS" sidebar has "GAS LIMIT" set to 600000, "VALUE" set to 0 Ether, and the "CONTRACT" selected is "bank_m - contracts/EtherBankUnderUnknownDeposit.sol". The "Deploy" button is orange and says "0x5B38Da6a701c568545dCf". Below it is a "Publish to IPFS" checkbox. The "Transactions recorded" section shows 111 entries, with a checkbox for "Run transactions using the latest compilation result" and "Save" and "Run" buttons. The "Deployed/Unpinned Contracts" section shows a deployed contract "BANK_M AT 0X2AC...75F3C (MEMC)". Underneath it, the balance is 4 ETH, and there are "deposit" and "withdraw" buttons. On the right, the code editor displays the Solidity code for "rbanckA2.sol" and "EtherBankUnderUnknownDeposit.sol". The "EtherBankUnderUnknownDeposit.sol" code includes logic for depositing and withdrawing ether from Alice and Bob's accounts. A transaction log at the bottom indicates a successful deposit from Alice's address.

```

//Riley.Thompson
//826526487
pragma solidity ^ 0.4.26;
contract bank_m {
    address private AliceA;
    address private BobB;
    uint private AliceBalance;
    uint private BobBalance;
    uint private totalBalance;
    constructor(address Alice, address Bob) public {
        AliceA = Alice;
        BobB = Bob;
    }
    // for new solidity, fallback() didn't work and compiler failed,
    // but this is the fallback function and does the same thing
    // changed fallback() to function()
    // ensures that Charlie can't send a basic Ethereum transaction that calls no function in bank_m.
    function() external payable {
        require(msg.sender == AliceA || msg.sender == BobA, "Fail!");
        if (msg.sender == AliceA) {
            AliceBalance += msg.value;
        } else if (msg.sender == BobA) {
            BobBalance += msg.value;
        }
    }
    function deposit() public payable {
        // Only Alice and Bob are able to use, otherwise fails
        require(msg.sender == AliceA || msg.sender == BobA, "Fail!");
        if (msg.sender == AliceA) {
            AliceBalance += msg.value;
        } else if (msg.sender == BobA) {
            BobBalance += msg.value;
        }
        totalBalance += msg.value;
    }
    function withdraw(uint amount) public {
    }
}

```

But can't with Charlie address:

This screenshot shows the same Truffle UI setup as the first one, but with a different outcome. The transaction log now shows a failure from Charlie's address, with the message "transact to bank_m.deposit errored: Error occurred: revert". The revert message details that the transaction was reverted to the initial state due to a "Fail!" reason provided by the contract.

```

//Riley.Thompson
//826526487
pragma solidity ^ 0.4.26;
contract bank_m {
    address private AliceA;
    address private BobB;
    uint private AliceBalance;
    uint private BobBalance;
    uint private totalBalance;
    constructor(address Alice, address Bob) public {
        AliceA = Alice;
        BobB = Bob;
    }
    // for new solidity, fallback() didn't work and compiler failed,
    // but this is the fallback function and does the same thing
    // changed fallback() to function()
    // ensures that Charlie can't send a basic Ethereum transaction that calls no function in bank_m.
    function() external payable {
        require(msg.sender == AliceA || msg.sender == BobA, "Fail!");
        if (msg.sender == AliceA) {
            AliceBalance += msg.value;
        } else if (msg.sender == BobA) {
            BobBalance += msg.value;
        }
    }
    function deposit() public payable {
        // Only Alice and Bob are able to use, otherwise fails
        require(msg.sender == AliceA || msg.sender == BobA, "Fail!");
        if (msg.sender == AliceA) {
    }
}

```

Here is my code in a picture:

DEPLOY & RUN TRANSACTIONS ✓ >

ACCOUNT 0x5B3...eddC4 (41.999999)

GAS LIMIT 600000

VALUE 0 Ether

CONTRACT bank_m - contracts/EtherBankUnderUnknownDeposit.sol

evm version: byzantium

Deploy 0x5B38Da6a701c568545dCf

Publish to IPFS

At Address Load contract from Address

Transactions recorded 111 ⓘ

Run transactions using the latest compilation result

Save **Run**

Deployed/Unpinned Contracts

Currently you have no unpinned contracts to interact with.

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.4.26;
3
4 contract bank_m {
5     address private Alice;
6     address private Bob;
7
8     uint private AliceBalance;
9     uint private BobBalance;
10    uint private totalBalance;
11
12    constructor(address Alice, address Bob) public {
13        AliceA = Alice;
14        BobB = Bob;
15    }
16
17    // for now solidity's fallback() didn't work and compiler failed,
18    // but this is the fallback function and does the same thing
19    // changed fallback() to function()
20    // ensures that Charlie can't send a basic Ethereum transaction that calls no function in bank_m.
21    function() external payable {
22        require(msg.sender == AliceA || msg.sender == BobB, "Fail");
23        if (msg.sender == AliceA) {
24            AliceBalance += msg.value;
25        } else if (msg.sender == BobB) {
26            BobBalance += msg.value;
27        }
28    }
29
30    function deposit() public payable {
31        // Only Alice and Bob are able to use, otherwise fails
32        require(msg.sender == AliceA || msg.sender == BobB, "Fail");
33        if (msg.sender == AliceA) {
34            AliceBalance += msg.value;
35        } else if (msg.sender == BobB) {
36            BobBalance += msg.value;
37        }
38        totalBalance += msg.value;
39    }
40
41    function withdraw(uint amount) public {
42        require(msg.sender == AliceA || msg.sender == BobB, "Fail");
43        if (msg.sender == AliceA) {
44            require(AliceBalance == amount, "Fail");
45            AliceBalance -= amount;
46            totalBalance -= amount;
47            msg.sender.transfer(amount);
48        } else if (msg.sender == BobB) {
49            require(BobBalance == amount, "Fail");
50            BobBalance -= amount;
51            totalBalance -= amount;
52            msg.sender.transfer(amount);
53        }
54    }
55 }
```

0 Listen on all transactions ⌂ Filter with transaction hash or address ⌂

revert
The transaction has been reverted to the initial state.
Reason provided by the contract: "Fail".

The file that is submitted is EtherBankUnderUnknownDeposit.sol