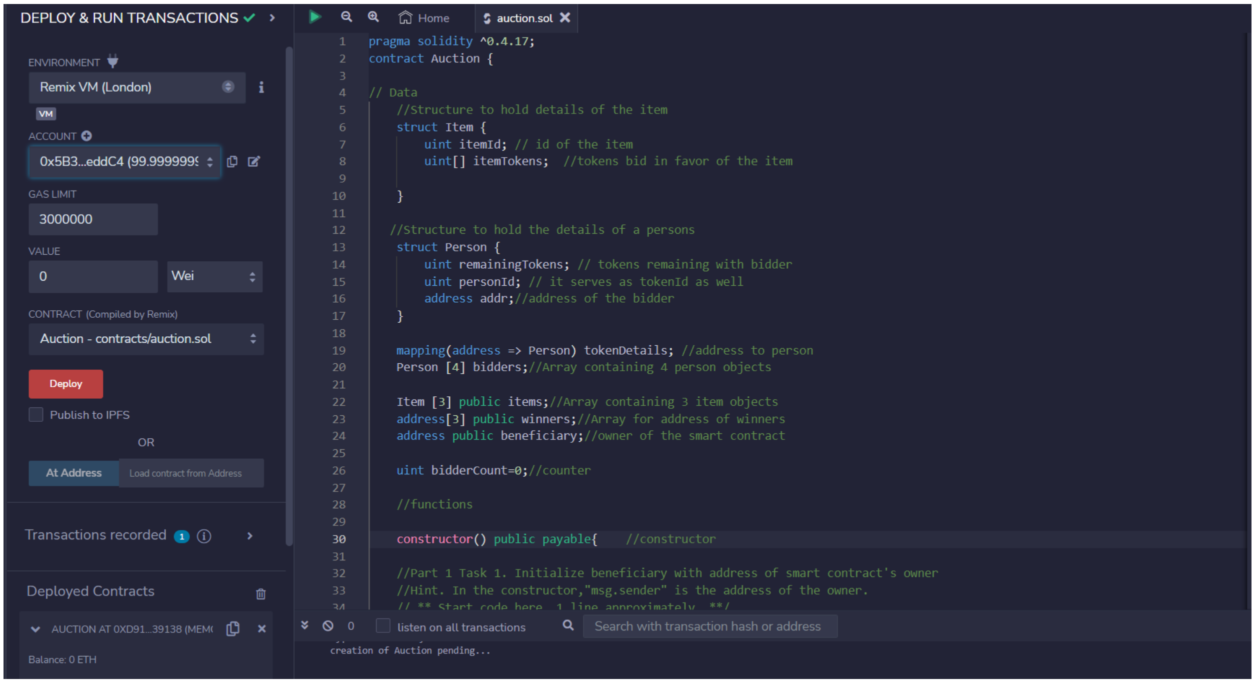
Introduction to Blockchains

Assignment 2

Devdatt N(200010012)

Version 1 :

Here, we have successfully deployed the contract after adding the requisite details:



DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Remix VM (London)

ACCOUNT: 0x5B3...eddC4 (99.9999999

GAS LIMIT: 3000000

VALUE: 0 Wei

CONTRACT (Compiled by Remix): Auction - contracts/auction.sol

□ Publish to IPFS

Transactions recorded 1

Deployed Contracts

AUCTION AT 0XD91...39138 (MEM(

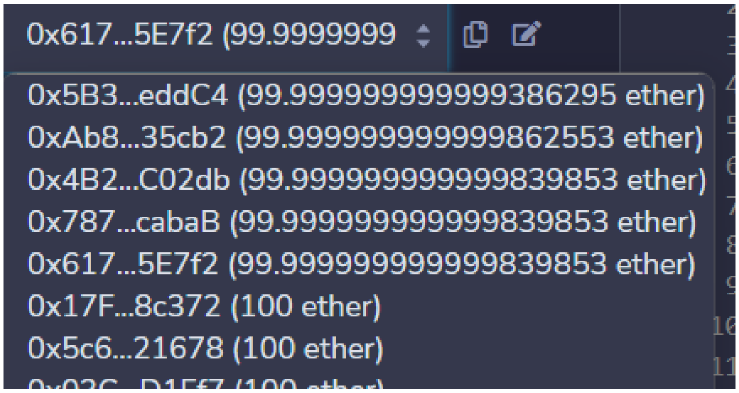
Balance: 0 ETH

```
pragma solidity ^0.4.17;
contract Auction {

// Data
    //Structure to hold details of the item
    struct Item {
        uint itemId; // id of the item
        uint[] itemTokens;  //tokens bid in favor of the item

    }

   //Structure to hold the details of a persons
    struct Person {
        uint remainingTokens; // tokens remaining with bidder
        uint personId; // it serves as tokenId as well
        address addr;//address of the bidder
    }

    mapping(address => Person) tokenDetails; //address to person
    Person [4] bidders;//Array containing 4 person objects

    Item [3] public items;//Array containing 3 item objects
    address[3] public winners;//Array for address of winners
    address public beneficiary;//owner of the smart contract

    uint bidderCount=0;//counter

    //functions

    constructor() public payable{    //constructor

    //Part 1 Task 1. Initialize beneficiary with address of smart contract's owner
    //Hint. In the constructor,"msg.sender" is the address of the owner.
    // ** Start code here. 1 line approximately. **/
```

```
creation of Auction pending...
```

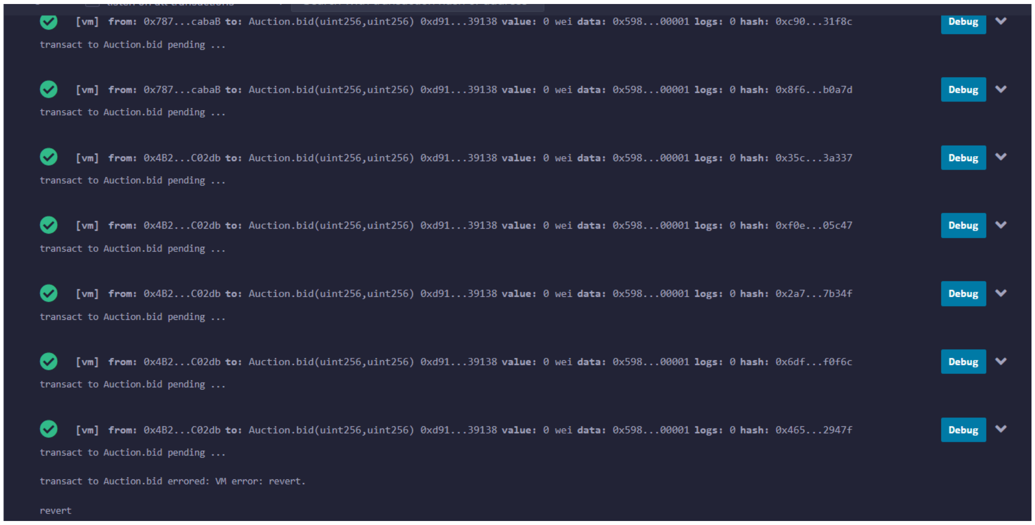
Deploy
OR
At Address
Load contract from Address
Home
auction.sol
0
listen on all transactions
Search with transaction hash or address

Now, we will register with all four accounts. Account balances after registering the four bidding accounts:



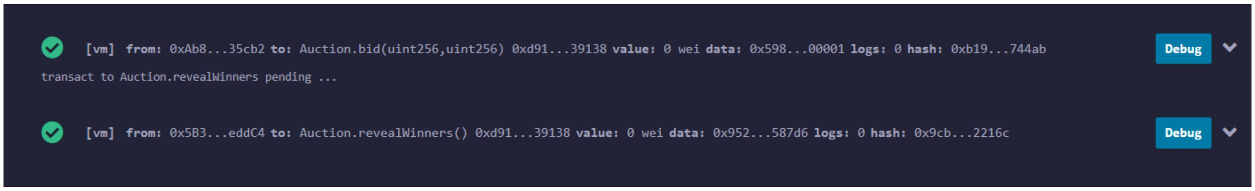
```
0x617...5E7f2 (99.9999999
0x5B3...eddC4 (99.999999999999386295 ether)
0xAb8...35cb2 (99.999999999999862553 ether)
0x4B2...C02db (99.999999999999839853 ether)
0x787...cabaB (99.999999999999839853 ether)
0x617...5E7f2 (99.999999999999839853 ether)
0x17F...8c372 (100 ether)
0x5c6...21678 (100 ether)
0x03C...D1Ff7 (100 ether)
```

Executing bid function for each of the four accounts: \



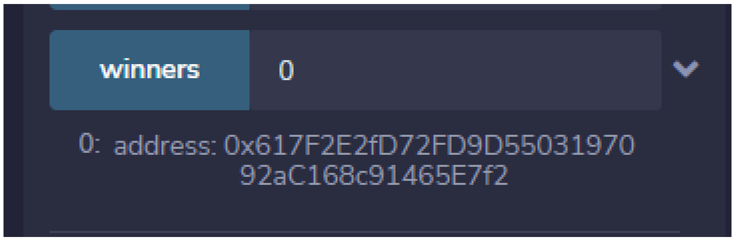
```
[vm] from: 0x787...cabaB to: Auction.bid(uint256,uint256) 0xd91...39138 value: 0 wei data: 0x598...00001 logs: 0 hash: 0xc90...31f8c
transact to Auction.bid pending ...

[vm] from: 0x787...cabaB to: Auction.bid(uint256,uint256) 0xd91...39138 value: 0 wei data: 0x598...00001 logs: 0 hash: 0x8f6...b0a7d
transact to Auction.bid pending ...

[vm] from: 0x4B2...C02db to: Auction.bid(uint256,uint256) 0xd91...39138 value: 0 wei data: 0x598...00001 logs: 0 hash: 0x35c...3a337
transact to Auction.bid pending ...

[vm] from: 0x4B2...C02db to: Auction.bid(uint256,uint256) 0xd91...39138 value: 0 wei data: 0x598...00001 logs: 0 hash: 0xf0e...05c47
transact to Auction.bid pending ...

[vm] from: 0x4B2...C02db to: Auction.bid(uint256,uint256) 0xd91...39138 value: 0 wei data: 0x598...00001 logs: 0 hash: 0x2a7...7b34f
transact to Auction.bid pending ...

[vm] from: 0x4B2...C02db to: Auction.bid(uint256,uint256) 0xd91...39138 value: 0 wei data: 0x598...00001 logs: 0 hash: 0x6df...f0f6c
transact to Auction.bid pending ...

[vm] from: 0x4B2...C02db to: Auction.bid(uint256,uint256) 0xd91...39138 value: 0 wei data: 0x598...00001 logs: 0 hash: 0x465...2947f
transact to Auction.bid pending ...

transact to Auction.bid errored: VM error: revert.

revert
```

Now, we execute revealWinners() :

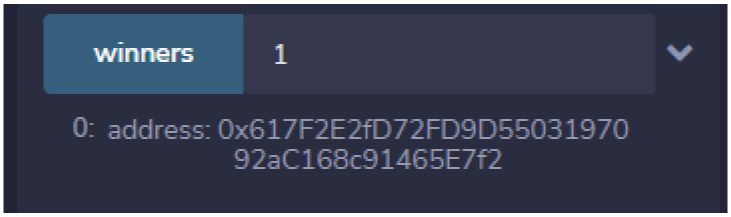


```
[vm] from: 0xAb8...35cb2 to: Auction.bid(uint256,uint256) 0xd91...39138 value: 0 wei data: 0x598...00001 logs: 0 hash: 0xb19...744ab
transact to Auction.revealWinners pending ...

[vm] from: 0x5B3...eddC4 to: Auction.revealWinners() 0xd91...39138 value: 0 wei data: 0x952...587d6 logs: 0 hash: 0x9cb...2216c
```

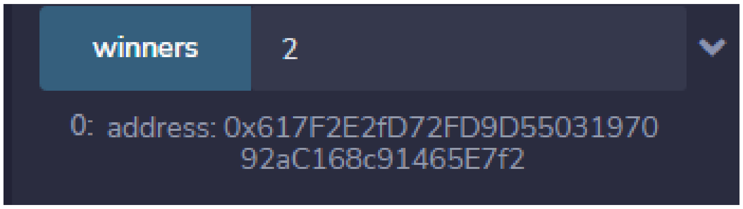
After this, we can get the winners:



```
winners 0
0: address: 0x617F2E2fD72FD9D55031970
            92aC168c91465E7f2
```



```
winners 1
0: address: 0x617F2E2fD72FD9D55031970
            92aC168c91465E7f2
```

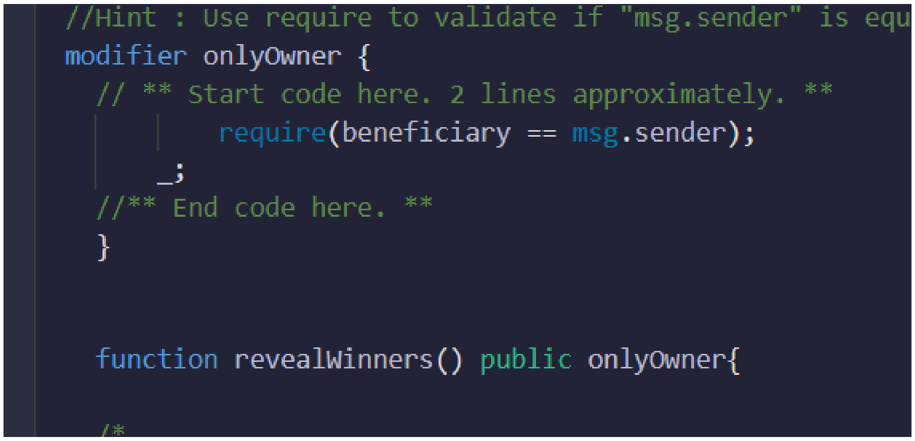


```
winners 2
0: address: 0x617F2E2fD72FD9D55031970
            92aC168c91465E7f2
```

We can see that the winner is the same in all three cases. This might be because the random value we generate is in fact not random.

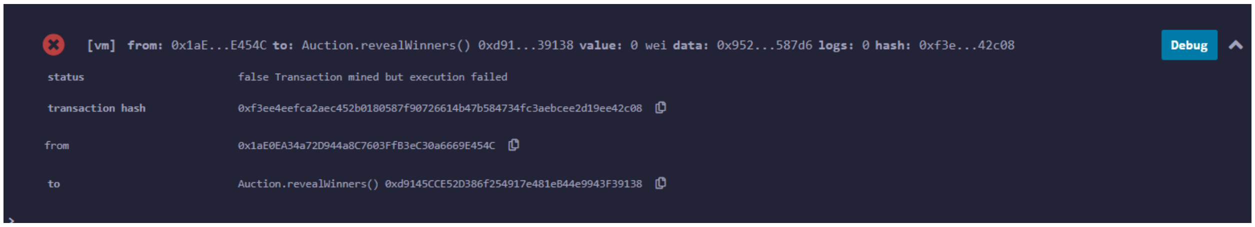
Version 2 :

We have added the onlyOwner modifier to revealWinners() so that only the auctioneer can call the function:



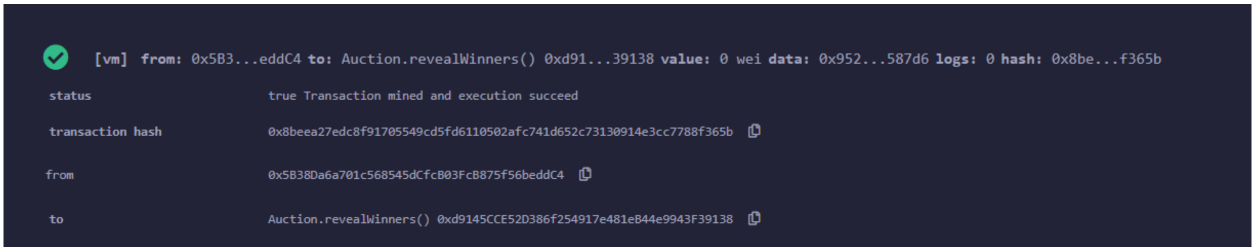
```
//Hint : Use require to validate if "msg.sender" is equ
modifier onlyOwner {
  // ** Start code here. 2 lines approximately. **
          require(beneficiary == msg.sender);
      _;
  //** End code here. **
  }


  function revealWinners() public onlyOwner{

  /*
```

In this case, after trying to execute revealWinners() from an address that is not the auctioneer, we come across an error:



```
[vm] from: 0x1aE...E454C to: Auction.revealWinners() 0xd91...39138 value: 0 wei data: 0x952...587d6 logs: 0 hash: 0xf3e...42c08
status              false Transaction mined but execution failed
transaction hash    0xf3ee4eefca2aec452b0180587f90726614b47b584734fc3aebcee2d19ee42c08
from                0x1aE0EA34a72D944a8C7603FfB3eC30a6669E454C
to                  Auction.revealWinners() 0xd9145CCE52D386f254917e481eB44e9943F39138
```

However, on calling the function from the auctioneer's account:



```
[vm] from: 0x5B3...eddC4 to: Auction.revealWinners() 0xd91...39138 value: 0 wei data: 0x952...587d6 logs: 0 hash: 0x8be...f365b
status              true Transaction mined and execution succeed
transaction hash    0x8beea27edc8f91705549cd5fd6110502afc741d652c73130914e3cc7788f365b
from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                  Auction.revealWinners() 0xd9145CCE52D386f254917e481eB44e9943F39138
```

We can see that the transaction is successful, therefore our smart contract is functioning.

Note: Version 1 smart contract code is Auction_1.sol, while Version 2 code is Auction_2.sol