Internal Practical

AIM:

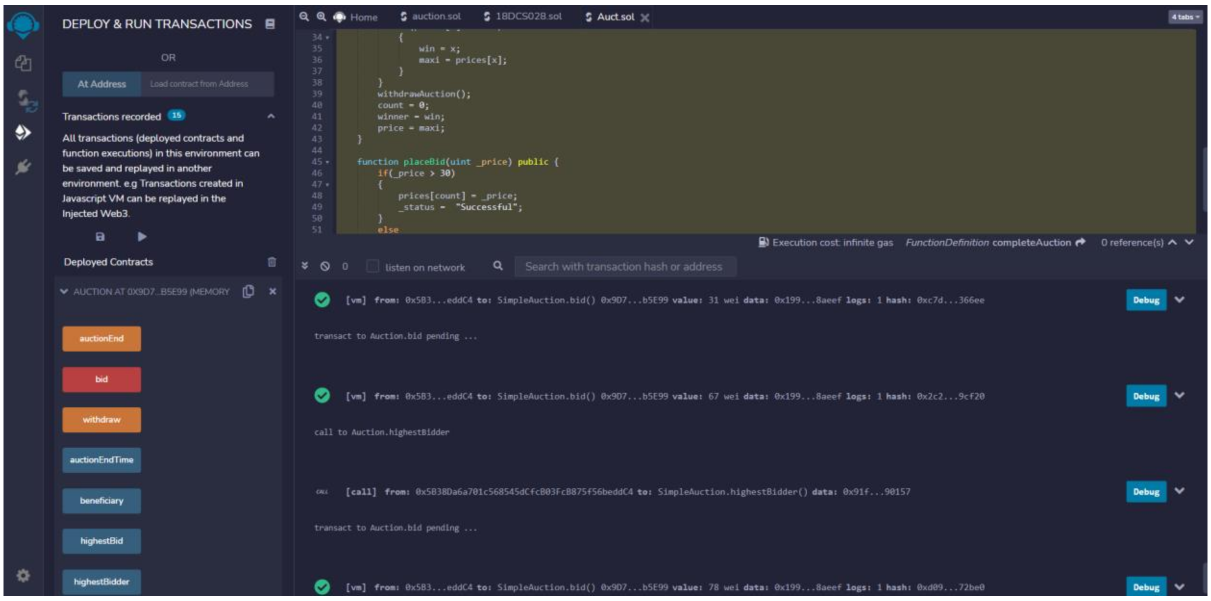
Implement a smart contract for Auction Application which have constructor that initialize min bid to 30 ethers. Also implement place bid function which will allow the customer to place bid. Also implement cancel auction which will cancel the auction. Implement withdraw and complete auction function. The withdraw function will be called once auction will be cancelled by the owner. The complete function will display the highest bid at the time of calling. Compile and Test the smart contract with different test cases and show the output. Admit at least 3 customers and show the output.

OUTPUT:

```
contract Auction {
   uint[] public prices = new uint[](10);
   uint public count = 0;
   uint public winner;
   uint public price;
   string public cancelled;
   string public _status;

   constructor() {
      cancelled = "";
      prices[0] = 30;
   }

   function cancelAuction () public {
      withdrawAuction();
      count = 0;
      cancelled = "Auction is cancelled";
   }

   function withdrawAuction() public {
      prices = new uint[](10);
      prices[0] = 30;
   }

   function completeAuction() public {
      uint maxi = 0;
      uint win;
      for(uint x = 0; x < prices.length; x++)
      {
         if(prices[x] > maxi)
         {
            win = x;
            maxi = prices[x];
         }
      }
      withdrawAuction();
      count = 0;
```

```
        winner = win;
        price = maxi;
    }

    function placeBid(uint _price) public {
        if(_price > 30)
        {
            prices[count] = _price;
            _status =  "Successful";
        }
        else
        {
            _status =  "At least 30 price is reuired";
        }
        count++;
    }
}
```

OUTPUT



DEPLOY & RUN TRANSACTIONS

OR

At Address | Load contract from Address

Transactions recorded 15

All transactions (deployed contracts and function executions) in this environment can be saved and replayed in another environment. e.g Transactions created in Javascript VM can be replayed in the Injected Web3.

Deployed Contracts

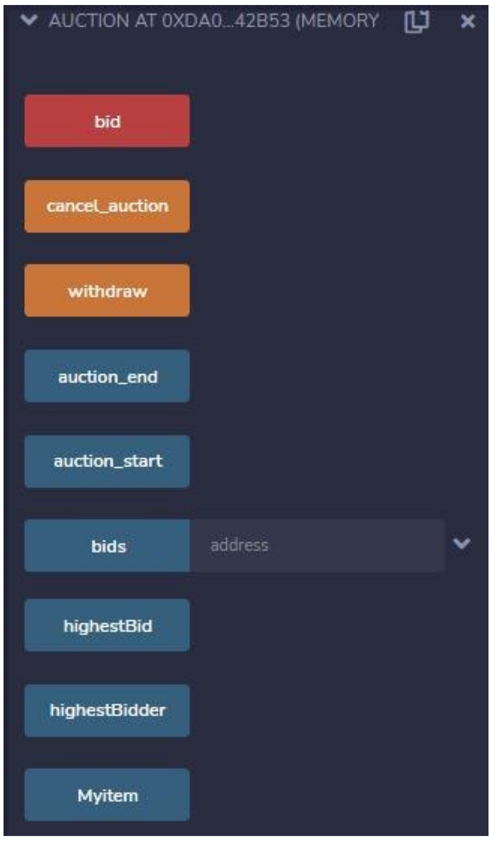
AUCTION AT 0X9D7...B5E99 (MEMORY

```
            {
                win = x;
                maxi = prices[x];
            }
        }
        withdrawAuction();
        count = 0;
        winner = win;
        price = maxi;
    }

    function placeBid(uint _price) public {
        if(_price > 30)
        {
            prices[count] = _price;
            _status =  "Successful";
        }
        else
```

Execution cost: infinite gas *FunctionDefinition* completeAuction 0 reference(s)

listen on network | Search with transaction hash or address

```
[vm] from: 0x5B3...eddC4 to: SimpleAuction.bid() 0x9D7...b5E99 value: 31 wei data: 0x199...8aeef logs: 1 hash: 0xc7d...366ee

transact to Auction.bid pending ...

[vm] from: 0x5B3...eddC4 to: SimpleAuction.bid() 0x9D7...b5E99 value: 67 wei data: 0x199...8aeef logs: 1 hash: 0x2c2...9cf20

call to Auction.highestBidder

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: SimpleAuction.highestBidder() data: 0x91f...90157

transact to Auction.bid pending ...

[vm] from: 0x5B3...eddC4 to: SimpleAuction.bid() 0x9D7...b5E99 value: 78 wei data: 0x199...8aeef logs: 1 hash: 0xd09...72be0
```



AUCTION AT 0XDA0...42B53 (MEMORY
bid
cancel_auction
withdraw
auction_end
auction_start
bids
address
highestBid
highestBidder
Myitem