



NATIONAL UNIVERSITY
of Computer & Emerging Sciences

Name: Usama Yousuf Khan

Sec: 7A

Roll no 20P-0646

Course: Block-Chain

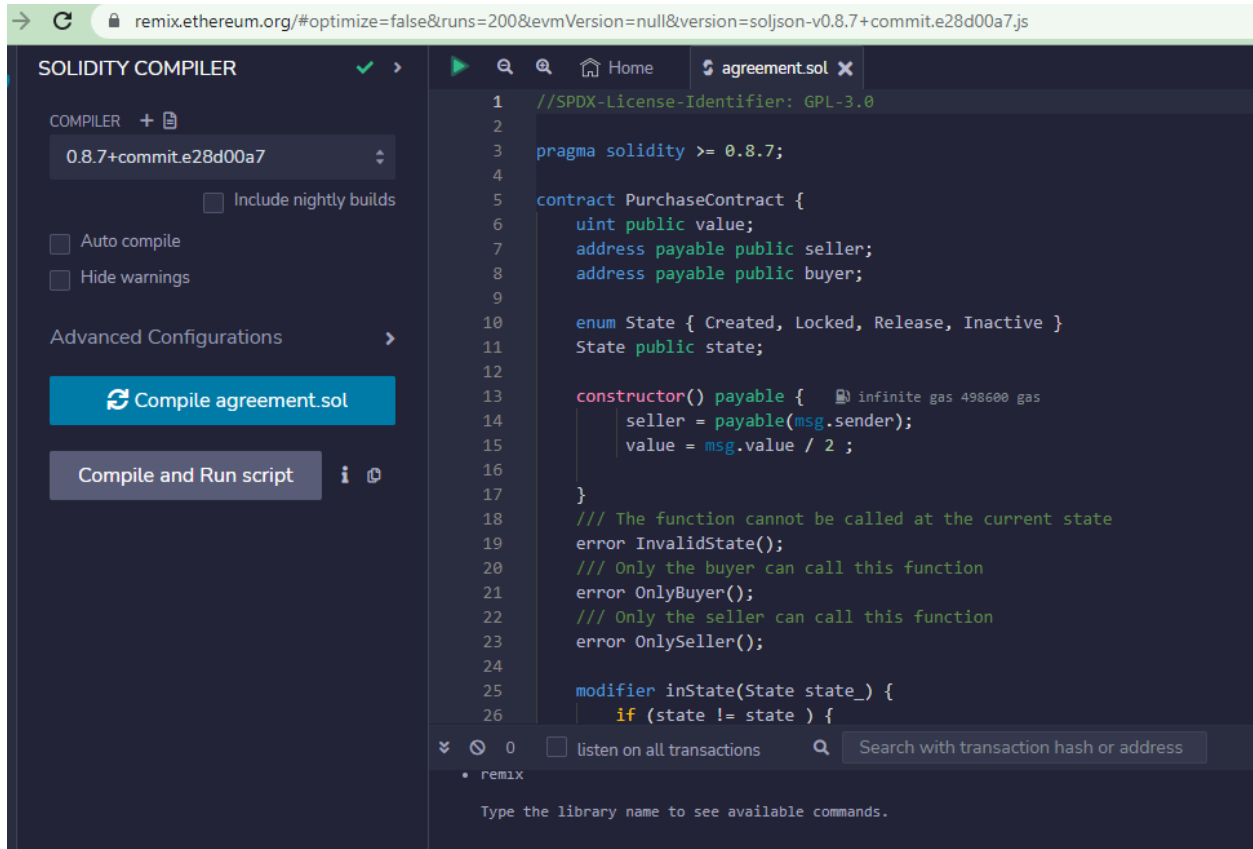
Project

Purchase Contract

Submitted to Bahar Ali

Screen Shots

Before Compilation

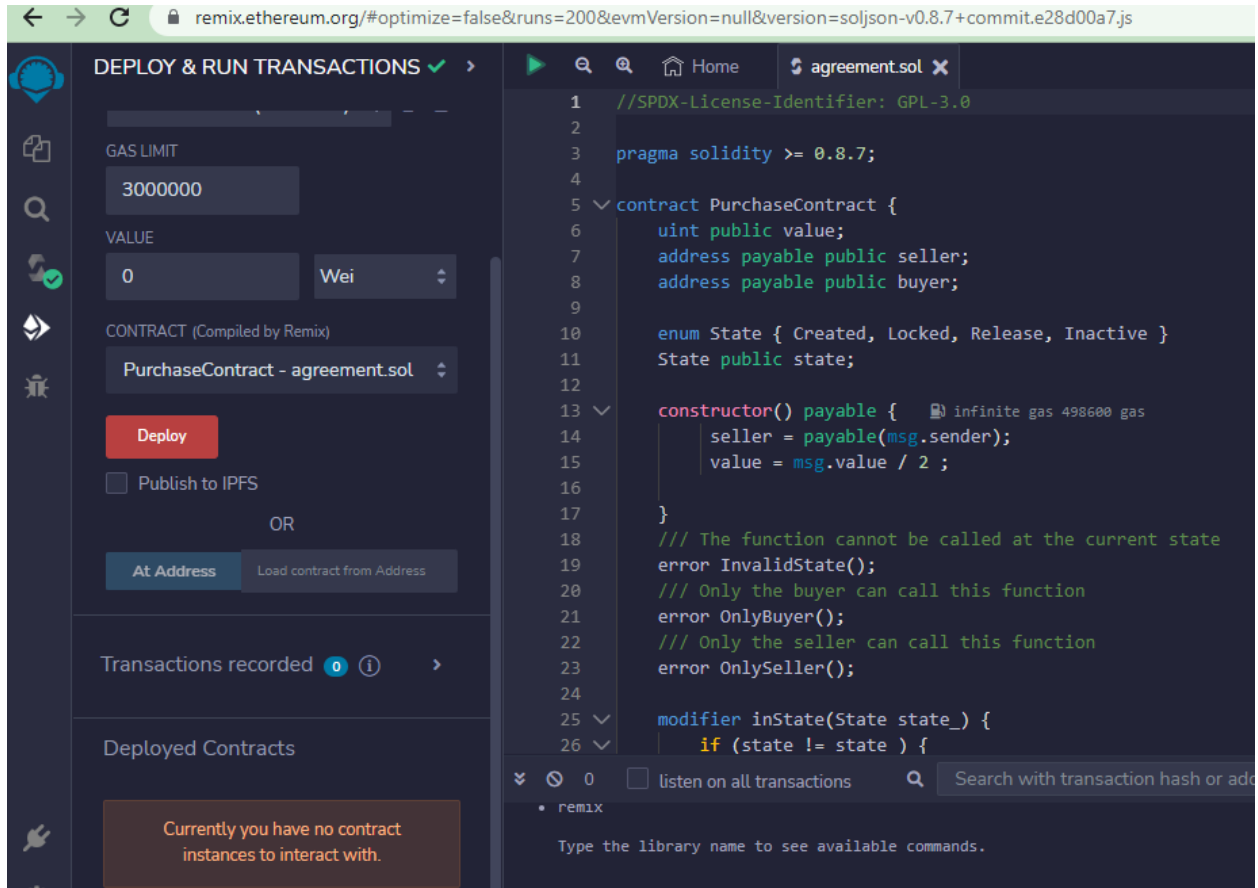


After Compilation:

The screenshot displays the Solidity Compiler web interface. On the left, the 'SOLIDITY COMPILER' sidebar shows the compiler version '0.8.7+commit.e28d00a7' and options for 'Include nightly builds', 'Auto compile', and 'Hide warnings'. The 'CONTRACT' dropdown is set to 'PurchaseContract (agreement.sol)'. Buttons for 'Compile agreement.sol', 'Compile and Run script', 'Publish on Ipfs', 'Publish on Swarm', and 'Compilation Details' are visible. The main editor shows the Solidity code for 'agreement.sol' with line numbers 1 through 26. The code includes a license header, pragma statement, contract definition, constructor, error definitions, and a modifier. The bottom status bar shows '0' transactions and a search bar. The Windows taskbar is at the bottom.

```
1 //SPDX-License-Identifier: GPL-3.0
2
3 pragma solidity >= 0.8.7;
4
5 contract PurchaseContract {
6     uint public value;
7     address payable public seller;
8     address payable public buyer;
9
10    enum State { Created, Locked, Release, Inactive }
11    State public state;
12
13    constructor() payable { infinite gas 498600 gas
14        seller = payable(msg.sender);
15        value = msg.value / 2 ;
16    }
17
18    /// The function cannot be called at the current state
19    error InvalidState();
20    /// Only the buyer can call this function
21    error OnlyBuyer();
22    /// Only the seller can call this function
23    error OnlySeller();
24
25    modifier inState(State state_) {
26        if (state != state_) {
```

Before Deployment



The screenshot displays the Remix IDE interface for deploying a smart contract. The left sidebar contains the 'DEPLOY & RUN TRANSACTIONS' panel, which includes fields for 'GAS LIMIT' (3000000) and 'VALUE' (0 Wei). Below these is a 'CONTRACT' dropdown menu showing 'PurchaseContract - agreement.sol'. A red 'Deploy' button is visible, along with a checkbox for 'Publish to IPFS'. The right sidebar shows the Solidity code editor with the following code:

```
1 //SPDX-License-Identifier: GPL-3.0
2
3 pragma solidity >= 0.8.7;
4
5 contract PurchaseContract {
6     uint public value;
7     address payable public seller;
8     address payable public buyer;
9
10    enum State { Created, Locked, Release, Inactive }
11    State public state;
12
13    constructor() payable {
14        seller = payable(msg.sender);
15        value = msg.value / 2 ;
16    }
17
18    /// The function cannot be called at the current state
19    error InvalidState();
20    /// Only the buyer can call this function
21    error OnlyBuyer();
22    /// Only the seller can call this function
23    error OnlySeller();
24
25    modifier inState(State state_) {
26        if (state != state_) {
```

At the bottom of the interface, a message states: 'Currently you have no contract instances to interact with.'

After Deployment

The screenshot displays the Remix IDE interface during the deployment of a smart contract. The left sidebar contains the 'DEPLOY & RUN TRANSACTIONS' panel, which shows the account '0x5B3...eddC4 (93.999999%)', a gas limit of 3,000,000, and a value of 0 Ether. The contract selected is 'PurchaseContract - agreement.sol'. The 'Deploy' button is highlighted, and the transaction is labeled 'Deploy - transact (payable)'. Below this, there are options to 'Publish to IPFS' or 'At Address'. The bottom section of the sidebar shows 'Transactions recorded' and 'Deployed Contracts', with a link to 'PURCHASECONTRACT AT 0XD91...'. The main editor displays the Solidity code for 'agreement.sol', which includes functions for confirming purchase, confirming receipt, paying the seller, and aborting. The bottom status bar shows a successful transaction: '[vm] from: 0x5B3...eddC4 to: PurchaseContract.(constructor) value: 6000000000000000 wei data: 0x608...7 hash: 0xb94...f6cf3'.

```
41     }
42     _;
43 }
44
45 function confirmpurchase() external inState(State.Created) payable {
46     require(msg.value == (2 * value), "Please send in 2x the purchase amount ");
47     buyer = payable(msg.sender);
48     state = State.Locked;
49 }
50
51 function confirmreceived() external inState(State.Locked) {
52     state = State.Release;
53     buyer.transfer(value);
54 }
55
56 function paySeller() external onlySeller inState(State.Release) {
57     state = State.Inactive;
58     seller.transfer(3 * value);
59 }
60
61 function abort() external onlySeller inState(State.Created) {
62     state = State.Inactive;
63     seller.transfer(address(this).balance);
64 }
65
66 }
```

[vm] from: 0x5B3...eddC4 to: PurchaseContract.(constructor) value: 6000000000000000 wei data: 0x608...7
hash: 0xb94...f6cf3

[illegible]