

DEEPAK KUMAR CHAUDHARY**12010440****IIT Bhilai**

1. Create multiple accounts in the meta mask and perform the balance transfer between the various accounts.

Ans- STEP:-

- Firstly open metamask.io & create account 1,2,3...
- Click open(), enter the address, and then take the test ether into account.
- Go to A/C 1 Menu click on Send option. click option “transfer “ between my account .
- After that gas fee are add and click on “Confirm”
- Finally balance transfer from one account to another.

2. Write a solidity program to set variables and get variables.

The screenshot displays the Remix IDE interface. The main editor shows a Solidity contract named `Storage` with the following code:

```
1 // SPDX-License-Identifier: GPL-3.0
2
3 pragma solidity >=0.7.0 <0.9.0;
4
5 /**
6  * @title Storage
7  * @dev Store & retrieve value in a variable
8  * @custom:dev-run-script ./scripts/deploy_with_ethereum.js
9  */
10 contract Storage {
11
12     uint256 myVar;
13
14     function seVar(uint256 newVar) public {
15         myVar = newVar;
16     }
17
18     /**
19     * @dev Return value
20     * @return value of 'number'
21     */
22
23     function getVar() public view returns (uint256) {
24         return myVar;
25     }
26 }
```

The left sidebar shows the "ENVIRONMENT" section with "JavaScript VM (London)" selected. The "CONTRACT" section lists "Storage - contracts/1_Storage.sol". The "Deploy" button is visible. The "Transactions recorded" section shows 2 transactions. The "Deployed Contracts" section shows "STORAGE AT 0xD91...39138 (MEMOF)". The "Interactions" section shows a "seVar" function call with the value "10". The "Level interactions" section shows a "Transact" button.

The bottom status bar shows the execution cost: 2415 gas. The function definition "retrieve" has 1 reference. The transaction log shows "transact to Storage.seVar errored: Error encoding arguments: Error: invalid argument" and "transact to Storage.seVar pending ...". The console shows a successful transaction: "[vm] from: 0x5B3...eddC4 to: Storage.seVar(uint256) 0xd91...39138".

Caption

3. Write a solidity program to perform push and pop operations on the dynamic array.

The screenshot displays the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel shows the deployment of a contract named 'DynamicArray' at address '0xD91...39138'. The 'Deployed Contracts' section lists the contract with its address. Below this, the 'addData' function is called with the value '10', and the 'getData' function is called, returning '0: int256[]: 10'. The 'getLength' function is also called, returning '0: uint256: 1'. The 'getSum' function is called, returning '0: int256: 10'. The 'search' function is called with the value '10', returning '0: bool: true'. The 'Low level interactions' section shows the 'CALLDATA' field. On the right, the Solidity code for the 'DynamicArray' contract is displayed, including the SPDX license, pragma statement, and functions for adding data, getting data, getting length, getting sum, and searching.

```
1 // SPDX-License-Identifier: GPL-3.0
2 // Write a solidity program to perform push and pop operations on the dynamic array.
3 pragma solidity ^0.6.8;
4 contract DynamicArray{
5
6     // Declaring state variable
7     int[] private arr;
8
9     // Function to add data
10    // in dynamic array
11    function addData(int num) public
12    {
13        arr.push(num);
14    }
15
16    // Function to get data of
17    // dynamic array
18    function getData() public view returns(int[] memory)
19    {
20        return arr;
21    }
22
23    // Function to return length
24    // of dynamic array
25    function getLength() public view returns (uint)
26    {
27        return arr.length;
28    }
29
30    // Function to return sum of
31    // elements of dynamic array
32    function getSum() public view returns(int)
33    {
34        uint i;
35        int sum = 0;
36
37        for(i = 0; i < arr.length; i++)
38            sum = sum + arr[i];
39        return sum;
40    }
41
42    // Function to search an
43    // element in dynamic array
```

4. write a solidity program to set address with a mapping variable

Ans- Code is written bellow.

```
pragma solidity ^0.7.0;
contract LedgerBalance {
    mapping(address => uint) public balances;
    function updateBalance(uint newBalance) public {
        balances[msg.sender] = newBalance;
    }
}
contract Updater {
    function updateBalance() public returns (uint) {
        LedgerBalance ledgerBalance = new LedgerBalance();
        ledgerBalance.updateBalance(10);
        return ledgerBalance.balances(address(this));
    }
}
```

5. Write a solidity program to get the factorial of a number.

Ans-

```
pragma solidity ^0.4.19;

contract Factorial {
```

```
function fact(uint x) returns (uint y) {  
    if (x == 0) {  
        return 1;  
    }  
    else {  
        return x*fact(x-1);  
    }  
}
```

6. Write a solidity program to store information about a student (name, roll no., institute, Age) using a structure.

Ans-

```
pragma solidity ^0.7.0;
```

```
contract Student {  
    struct detail {  
        string institute;  
        string name;  
        uint roll;  
    }  
    detail student;  
  
    function setStudent() public {  
        student = detail( 'IIT', 'Deepak', 12010440);  
    }  
    function getroll() public view returns (uint) {  
        return student.roll;  
    }  
}
```

```
}
```

7. Write a smart contract using a solidity program to perform the balance transfer from the contract to other accounts.

Ans-

```
// SPDX-License-Identifier: MIT
```

```
pragma solidity ^0.7.0;
```

```
contract CommunityChest {
```

```
    function deposit(uint256 amount) payable public {
```

```
        require(msg.value == amount);
```

```
        // nothing else to do!
```

```
    }
```

```
    function getBalance() public view returns (uint256) {
```

```
        return address(this).balance;
```

```
    }
```

```
}
```

8. Write a smart contract using a solidity program to perform balance transfer with mapping and make sure only the owner can transfer the balance from the contract to other accounts

Ans- // SPDX-License-Identifier: MIT

```
pragma solidity ^0.6.8;
```

```
// Creating a contract
```

```
contract MyContract
```

```
{
```

```
    // Private state variable
```

```
    address private owner;
```

```
    // Defining a constructor
```

```
    constructor() public {
```

```
        owner=msg.sender;
```

```
    // Function to get
```

```
    // address of owner
```

```
    function getOwner(
```

```
) public view returns (address) {
```

```
return owner;
```

The screenshot displays the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel shows the contract 'MyContract' selected. The 'Deploy' button is highlighted. Below it, there are options to 'Publish to IPFS' or 'At Address'. The 'Transactions recorded' section shows one transaction. The 'Deployed Contracts' section shows the contract deployed at address '0xD91...39138'. Below this, there are buttons for 'getBalance' and 'getOwner'. The 'Low level interactions' section shows the call data for the 'getOwner' function.

The main editor shows the Solidity code for 'MyContract':

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.6.8;
3 // Creating a contract
4 contract MyContract
5 {
6     // Private state variable
7     address private owner;
8
9     // Defining a constructor
10    constructor() public{
11        owner=msg.sender;
12    }
13
14    // Function to get
15    // address of owner
16    function getOwner(
17    ) public view returns (address) {
18        return owner;
19    }
20
21    // Function to return
22    // current balance of owner
23    function getBalance(
24    ) public view returns(uint256){
25        return owner.balance;
26    }
27 }
28

```

The bottom panel shows a transaction log with a call to 'MyContract.getOwner()'.

```
}
```

```
// Function to return
```

```
// current balance of owner
```

```
function getBalance(
```

```
) public view returns(uint256){
```

```
    return owner.balance;
```

```
}
```

```
}
```


9. Write a solidity program to perform the exception handling and describe the details with screenshots.

Ans -

```
pragma solidity ^0.5.0;
```

```
contract Vendor {  
    address public seller;  
    modifier onlySeller() {  
        require(  
            msg.sender == seller,  
            "Only seller can call this."  
        );  
    }  
}
```

```
function sell(uint amount) public payable onlySeller {  
    if (amount > msg.value / 2 ether)  
        revert("Not enough Ether provided.");  
    // Perform the sell operation.  
}
```

DEPLOY & RUN TRANSACTIONS

Deploy

Publish to IPFS

OR

At Address

Load contract from Address

Transactions recorded 9

Transactions (deployed contracts and function executions) in this environment can be saved and replayed in another environment. e.g Transactions created in a browser VM can be replayed in the selected Web3.

Deployed Contracts

VENDOR AT 0XF8E...9FBE8 (MEMORY)

sell 10

seller

address: 0x00

Low level interactions

Transact

1_Storage.sol

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.5.0;
3
4 contract Vendor {
5     address public seller;
6     modifier onlySeller() {
7         require(
8             msg.sender == seller,
9             "Only seller can call this."
10        );
11    }
12
13    function sell(uint amount) public payable onlySeller {
14        if (amount > msg.value / 2 ether)
15            revert("Not enough Ether provided.");
16    }
17 }
18

```

0

listen on all transactions

Search with transaction hash or address

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Vendor.seller() data: 0x085...51a53

transact to Vendor.sell pending ...

transact to Vendor.sell errored: VM error: revert.

revert

The transaction has been reverted to the initial state.
Reason provided by the contract: "Only seller can call this."
Debug the transaction to get more information.

[vm] from: 0x5B3...eddC4 to: Vendor.sell(uint256) 0xf8e...9fBe8 value: 0 wei data: 0xe48...0000a
hash: 0x682...2c8d3

Caption

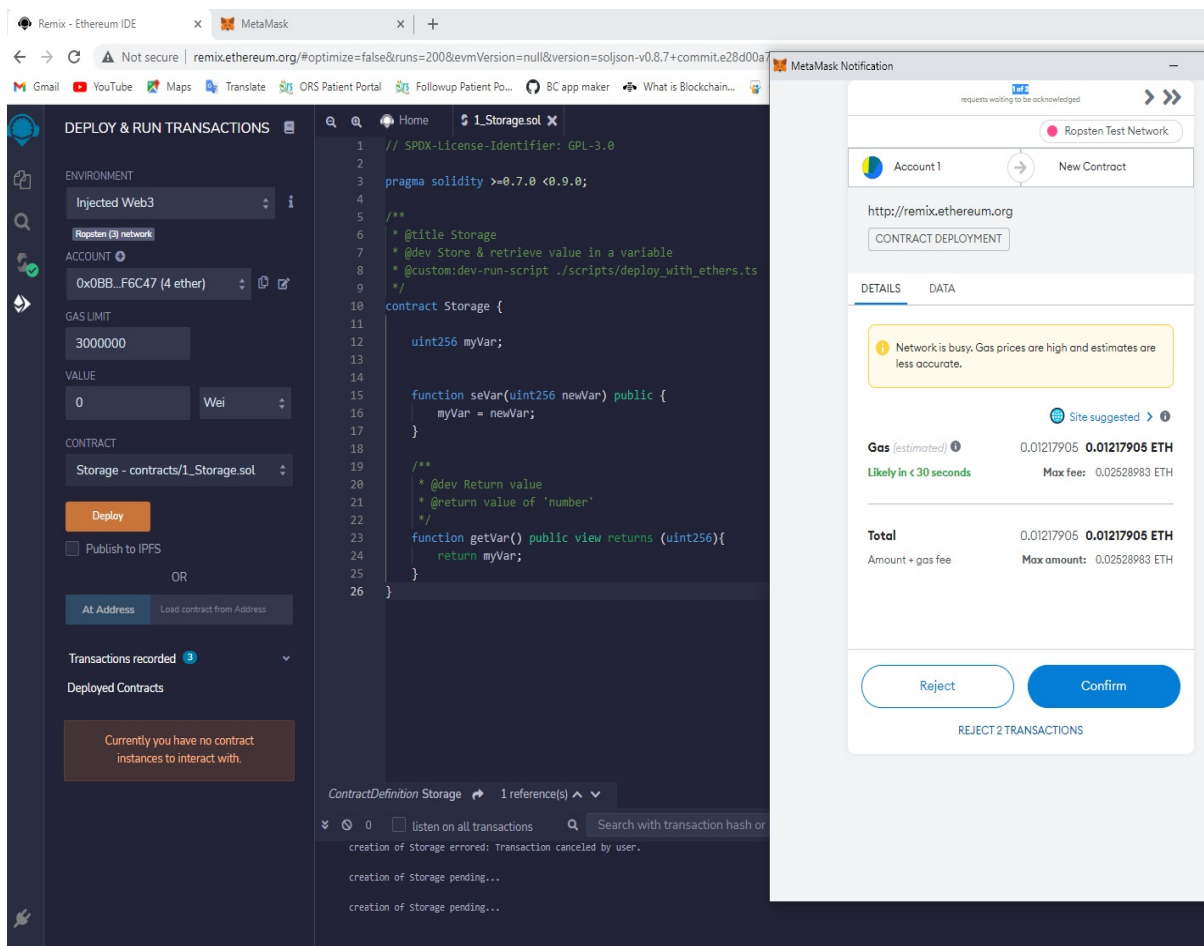
}

10

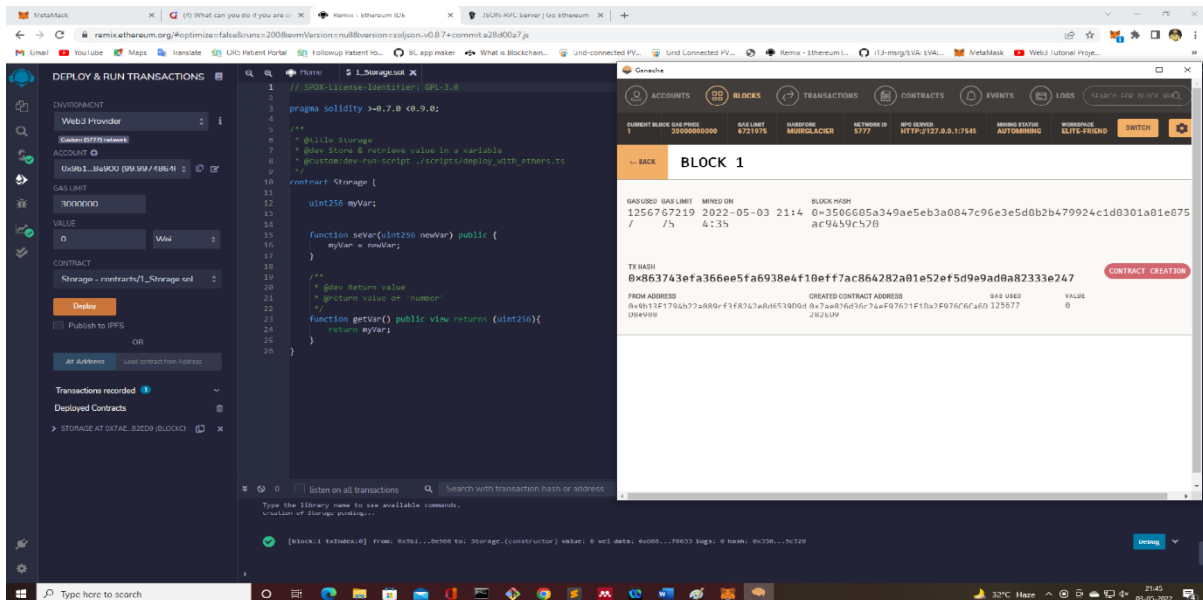
10. Connect the following tool with their mixed environment, perform balance transfer between the accounts with the smart contract and share the screenshots.

- Ganache
- Meta mask

Ans- firstly change the environment to “injected web3” and deploy the sol code. And after we get confirmation from meta mask shown in figure below.



-for connecting ganache, we change the environment to “web3 provider”, finally ganache is connected and we can see the transaction on Block wise shown in screenshot below.



*****END*****