Ethereum and Solidity lab assignments

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

Creating multiple accounts in MetaMask:

* In the MetaMask extension, click on the three dots in the upper right corner and select "Create Account."
* Repeat the process for creating additional accounts as needed.
* Performing balance transfer between the accounts in a blockchain:
* Make sure that the account you want to transfer funds from is selected in MetaMask.
* Input the destination account address and the amount of funds you want to transfer in the blockchain's transaction form.
* Confirm and sign the transaction using the selected account in MetaMask.

1. **Write a solidity program to set variable and get variable**

pragma solidity ^0.5.0;

contract Solidity\_var\_Test {

   uint8 public state\_var;

  constructor() public {

      state\_var = 16;

   }

}

1. **Write a solidity program to perform push and pop operations on dynamic array**

pragma solidity ^0.8.0;

contract DynamicArray {

uint[] array;

function push(uint value) public {

array.push(value);

}

function pop() public returns (uint) {

require(array.length > 0, "Array is empty, cannot pop");

uint value = array[array.length - 1];

array.length--;

return value;

}

}

1. **Write a solidity program to set address with a mapping variable.**

pragma solidity ^0.8.0;

contract AddressMapping {

mapping (address => uint) public balances;

function setAddress(address \_address, uint \_balance) public {

balances[\_address] = \_balance;

}

function getAddressBalance(address \_address) public view returns (uint) {

return balances[\_address];

}

}

1. **Write a solidity program to get the factorial of a number**

pragma solidity ^0.8.0;

contract Factorial {

function getFactorial(uint n) public view returns (uint) {

uint result = 1;

for (uint i = 1; i <= n; i++) {

result \*= i;

}

return result;

}

}

1. **Write a solidity program to store information of a student(Name,Rollno.,Institute,Age) using structure.**

pragma solidity ^0.8.0;

contract Student {

struct StudentInfo {

string name;

uint rollNo;

string institute;

uint age;

}

mapping (uint => StudentInfo) public students;

uint studentCount = 0;

function addStudent(string memory \_name, uint \_rollNo, string memory \_institute, uint \_age) public {

students[studentCount].name = \_name;

students[studentCount].rollNo = \_rollNo;

students[studentCount].institute = \_institute;

students[studentCount].age = \_age;

studentCount++;

}

function getStudent(uint \_index) public view returns (string memory, uint, string memory, uint) {

return (students[\_index].name, students[\_index].rollNo, students[\_index].institute, students[\_index].age);

}

}

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

pragma solidity ^0.8.0;

contract BalanceTransfer {

address payable public owner;

constructor() public {

owner = msg.sender;

}

function transfer(address payable \_to, uint256 \_value) public {

require(msg.sender == owner, "Only the owner can initiate a transfer");

require(\_to != address(0), "Invalid address");

require(\_value <= address(this).balance, "Insufficient balance");

\_to.transfer(\_value);

}

}

1. **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 contract to other accounts**

pragma solidity ^0.8.0;

contract BalanceTransfer {

address payable public owner;

mapping (address => uint256) public balances;

constructor() public {

owner = msg.sender;

balances[owner] = 1000000000000000000; // 10 ETH

}

function transfer(address payable \_to, uint256 \_value) public {

require(msg.sender == owner, "Only the owner can initiate a transfer");

require(\_to != address(0), "Invalid address");

require(balances[msg.sender] >= \_value, "Insufficient balance");

balances[msg.sender] -= \_value;

balances[\_to] += \_value;

}

}

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

pragma solidity ^0.8.0;

contract ExceptionHandling {

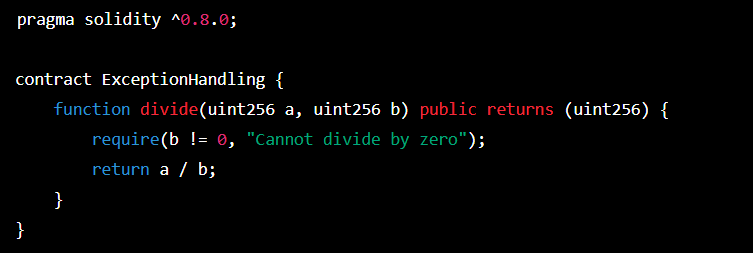
function divide(uint256 a, uint256 b) public returns (uint256) {

require(b != 0, "Cannot divide by zero");

return a / b;

}

}



1. **Connect the following tools with the remix environment and perform balance transfer between the accounts with smart contract and share the screen shots.**

**a) Ganache**

**b) Metamask**

To connect Ganache with the Remix environment, follow these steps:

1. Start Ganache and create a new workspace.
2. In Remix, click on the "Run" tab.
3. In the "Environment" drop-down menu, select "Web3 Provider".
4. In the "Web3 Provider Endpoint" field, enter the URL of the Ganache network, which is typically **http://127.0.0.1:7545**.
5. Click the "Connect" button.

To connect Metamask with the Remix environment, follow these steps:

1. Install Metamask browser extension.
2. Open Metamask and create a new wallet.
3. In Remix, click on the "Run" tab.
4. In the "Environment" drop-down menu, select "Web3 Provider".
5. In the "Web3 Provider Endpoint" field, enter the URL of the Metamask network, which is typically **http://127.0.0.1:7545**.
6. Click the "Connect" button.
7. In Metamask, switch to the network you're using in Remix and make sure that the Metamask account is unlocked.

To perform a balance transfer between accounts using a smart contract in Remix, follow these steps:

1. Write a smart contract to perform the balance transfer, as described in a previous answer.
2. In Remix, paste the smart contract code into the editor.
3. Compile the contract by clicking the "Compile" button.
4. In the "Run" tab, select the contract and click the "Deploy" button.
5. In Metamask, confirm the deployment transaction by clicking the "Confirm" button.
6. Interact with the deployed contract by calling its functions and making transactions. To transfer balance, call the **transfer** function and specify the recipient address and the transfer amount.
7. In Metamask, confirm the transfer transaction by clicking the "Confirm" button.

