9/03/2023

Symbiosis Skills and Professional University

Name: Kirti Prashant Khollam
PRN: 2001106075
Course code: CS601
Course: Cryptography and Blockchain

Practical 8

Aim: Functions in Solidity.

Apparatus:

- Remix IDE
- Lecture notes
- Computer
- Journals, etc.

Concept:

- A function is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing the same code again and again. It helps programmers in writing modular codes. Functions allow a programmer to divide a big program into a number of small and manageable functions.

```
function function-name(parameter-list) scope returns() {
  //statements
}
```

- View functions ensure that they will not modify the state. A function can be declared as view.
- Pure functions ensure that they not read or modify the state. A function can be declared as pure.
- Fallback function is a special function available to a contract. It has following features −
  - It is called when a non-existent function is called on the contract.
  - It is required to be marked external.
  - It has no name.
  - It has no arguments
  - It can not return any thing.
  - It can be defined one per contract.
  - If not marked payable, it will throw exception if contract receives plain ether without data.

Procedure and Observations:

```
pragma solidity 0.5.0;
// View functions ensure that they will not modify the state (return values)
// Pure functions ensure that they not read or modify the state (return calculations).
contract MyContract{
   uint value;
```

9/03/2023

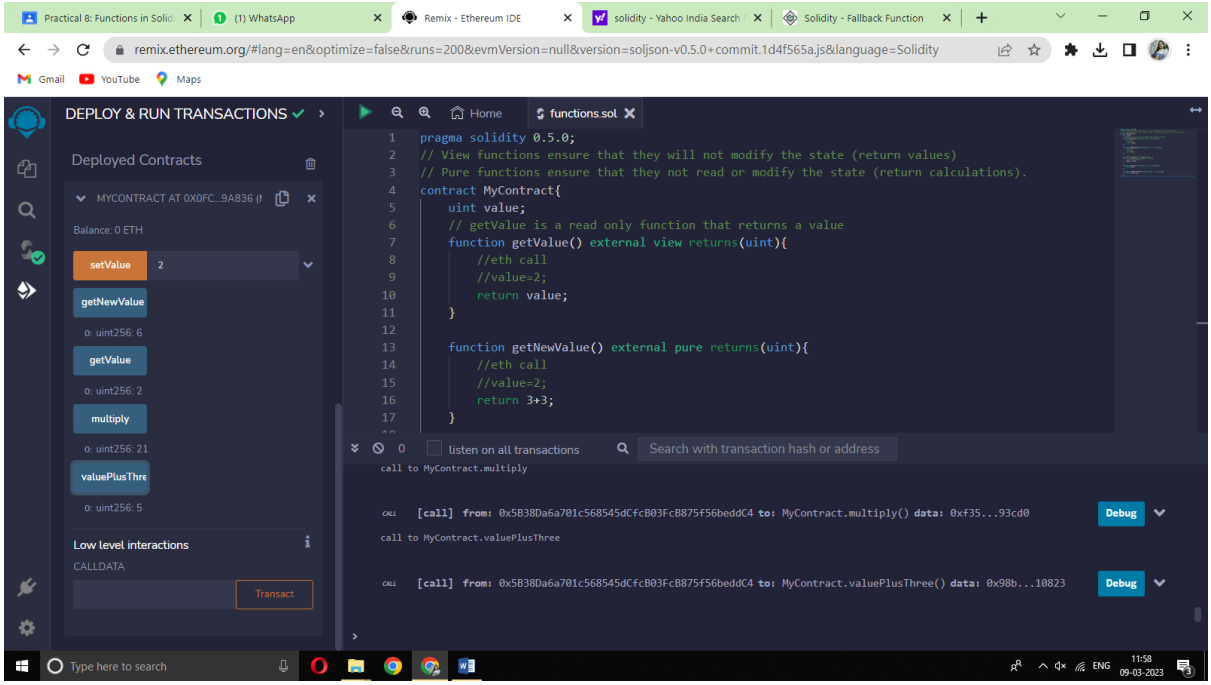
```
    // getValue is a read only function that returns a value
    function getValue() external view returns(uint){
        //eth call
        //value=2;
        return value;
    }

    function getNewValue() external pure returns(uint){
        //eth call
        //value=2;
        return 3+3;
    }

    // setValue modifies the state value
    function setValue(uint _value) external{
        //eth send transaction
        value= _value;
    }

    function multiply() external pure returns(uint){
        return 3*7;
    }

    function valuePlusThree() external view returns(uint){
        return value +3;
    }
}
```



9/03/2023

Functions Overloading:

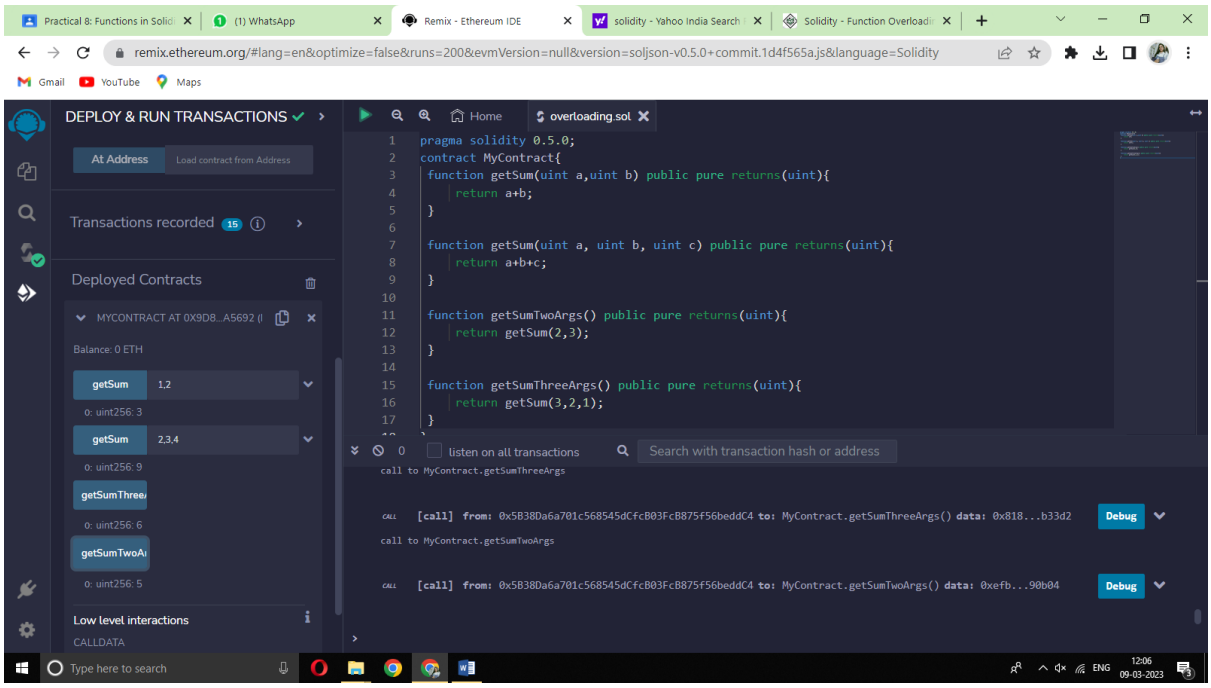
You can have multiple definitions for the same function name in the same scope. The definition of the function must differ from each other by the types and/or the number of arguments in the argument list. You cannot overload function declarations that differ only by return type.

```
pragma solidity 0.5.0;
contract MyContract{
 function getSum(uint a,uint b) public pure returns(uint){
    return a+b;
 }

 function getSum(uint a, uint b, uint c) public pure returns(uint){
    return a+b+c;
 }

 function getSumTwoArgs() public pure returns(uint){
    return getSum(2,3);
 }

 function getSumThreeArgs() public pure returns(uint){
    return getSum(3,2,1);
 }
}
```



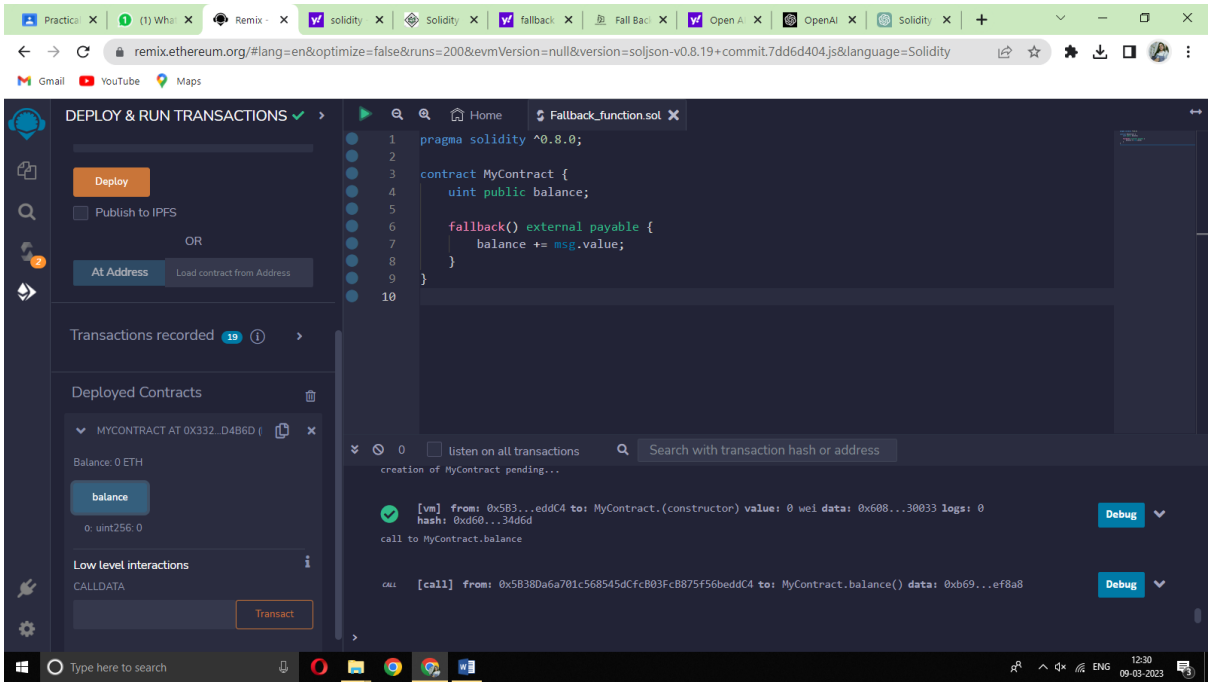
9/03/2023

Function Fallback:

```
pragma solidity ^0.8.0;

contract MyContract {
    uint public balance;

    fallback() external payable {
        balance += msg.value;
    }
}
```



Conclusion: Hence, I implemented functions in solidity.