Sr.No	Date	Practical Title	Sign
1	11.02.2023	Practical No 1 Create blockchain with 3 blocks and hence display the entire blockchain, hash value and timestamp of each block.	
2	17.02.2023	Practical No 2 Create a smart Contract and Implement & demonstrate the use of solidity programming 1. Create a smart contract for Counter 2. Create a smart contract for Increment and decrement operator	
3	17.03.2023	Practical No 3 Create a Smart Contract in solidity program to demonstrate array and its types.	
4	24.03.2023	Practical No 4 1. Solidity program to demonstrate Comparison operators. 2. Solidity program to demonstrate Logical operators. 3. Solidity program to demonstrate Assignment operators. 4. Solidity program to demonstrate Ternary operators. 5. Solidity program to demonstrate Bitwise operators.	
5	31.03.2023	Practical No 5 Create smart contract in loops using Solidity Programming 1. Create a Smart contract for loop 2. Create a Smart contract while loop	

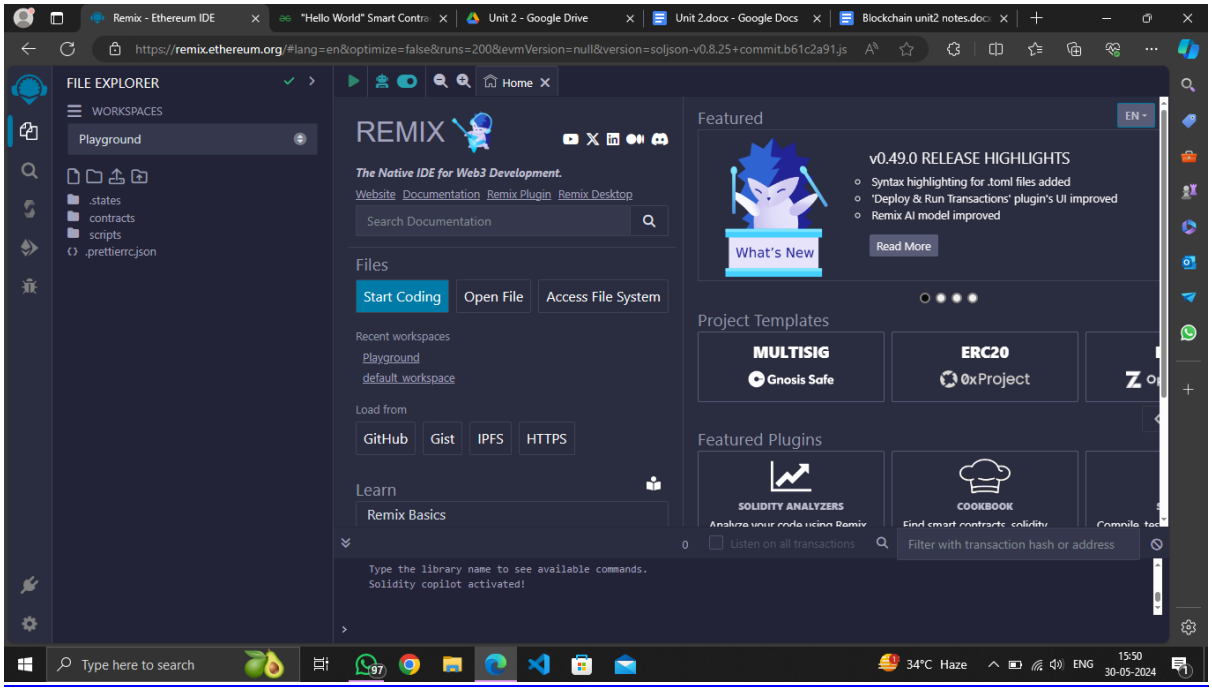
6	31.04.2023	Practical No 6 1. Create a smart contract to demonstrate Mathematical function. 2. Create a smart contract to demonstrate Function overloading.	
7	04.05.2023	Practical No 7 Create a Smart contract for 1.Implementation of Interface & 2.Inheritance	
8	08.05.2023	Practical No 8 Create smart contract for Selection of candidate in election.	
9	10.05.2023	Practical No 9 Write a solidity program to create an array of role no.& create a smart contract where it checks the value of roll no.s & perform AND operation with today's date DD and if the result is even display a message " Student is ALLOWED." else display "DENIED".	
10	11.05.2023	Practical No 10 Write a solidity program to find the sum of an array of ten numbers using loop the numbers are expected to be taken from the user, create a smart contract to find the AND operation of odd positioned numbers and OR operation of even positioned numbers including 0^{th} index. Hence find the product of the results and also identify whether the result is the part of array or not.	

We are using the Remix IDE to perform in Solidity Programming.

Steps to perform:

1. **Open Remix:**

Open any browser and type **https://remix.ethereum.org**

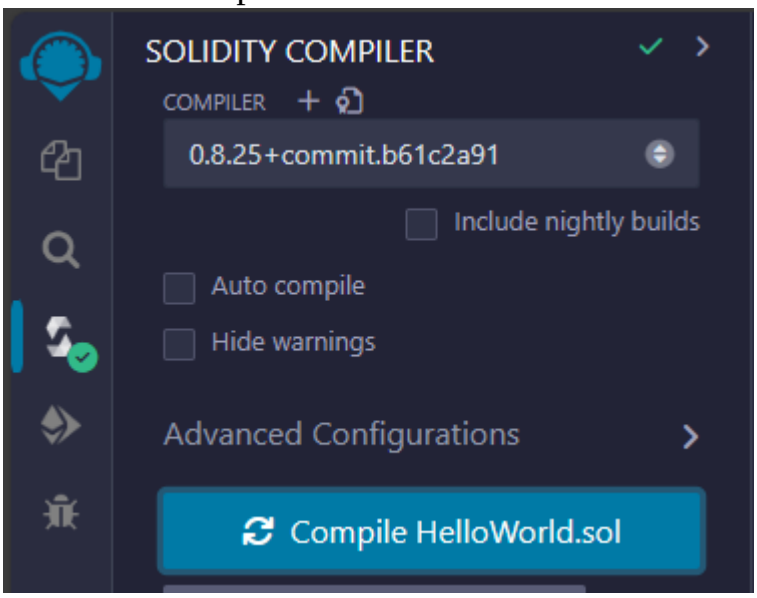


2. **Create New Files:**

- Click on the “file explorer” icon onto the left side bar (indicated by blue arrow in the above picture).
- Select Solidity in the Environment and click + symbol right to the browser.
- In the "File Explorer" pane, create new files by any name such as HelloWorld.sol. It will look like this:

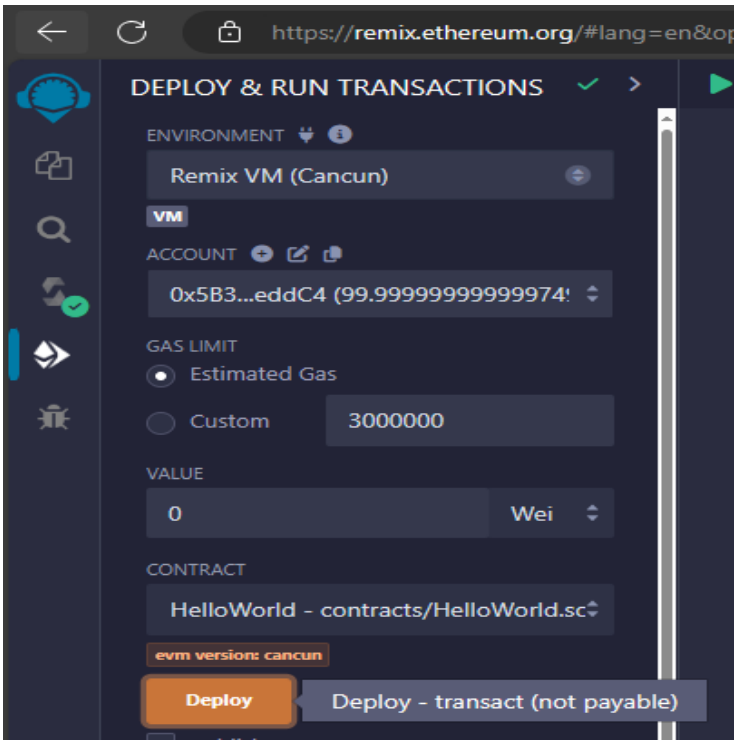
3. **Compile the Contracts:**

- Click on the "Solidity Compiler" tab.
- Ensure the appropriate compiler version (between 0.6.12 and 0.9.0) is selected.
- Click the "Compile" button for each contract.

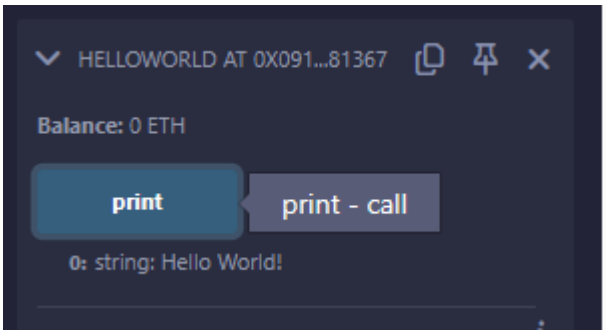


4. **Deploy the Contracts:**

- Click on the "Deploy & Run Transactions" tab.
- Ensure "JavaScript VM" is selected in the "Environment" dropdown for a local blockchain simulation.
- Deploy each contract by selecting it and clicking the "Deploy" button.



5. **Interact with the Contracts**



HelloWorld.sol

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.6.12 <0.9.0;

contract HelloWorld {
  /**
   * @dev Prints Hello World string
   */
  function print() public pure returns (string memory) {
    return "Hello World!";
  }
}
```

Solidity Programming

Solidity is an object-oriented and high-level language for implementing Smart contracts. Solidity is a curly-bracket language designed to target the Ethereum Virtual Machine(EVM), It is influenced by C++, Python, and JavaScript. Solidity is statically typed and supports inheritance, libraries, and complex user-defined types among features. With this, you can create contracts for users such as voting, crowdfunding, blind auctions, and multi-signature wallets.

SPDX License Identifier:

// SPDX-License-Identifier: MIT: This specifies the license under which the contract is distributed. MIT is a permissive open-source license.

Pragma Directive:

pragma solidity >=0.6.12 <0.9.0;: Specifies that the Solidity compiler version should be between 0.6.12 (inclusive) and 0.9.0 (exclusive).

Contract Definition: contract name { ... }: Defines the smart contract namee.

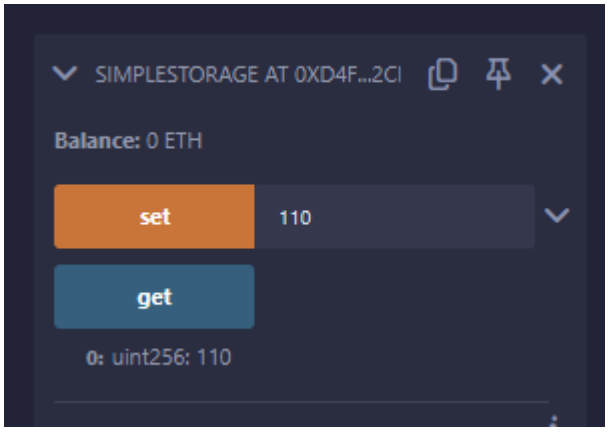
Practical No:1

Aim:- Create blockchain with 3 blocks and hence display the entire blockchain, hash value and timestamp of each block.

Code:

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.6.12 <0.9.0;

contract SimpleStorage {
    uint256 private storedData;

    event DataStored(uint256 data);

    function set(uint256 x) public {
        storedData = x;
        emit DataStored(x);
    }

    function get() public view returns (uint256) {
        return storedData;
    }
}
```

Output:



Practical No:2

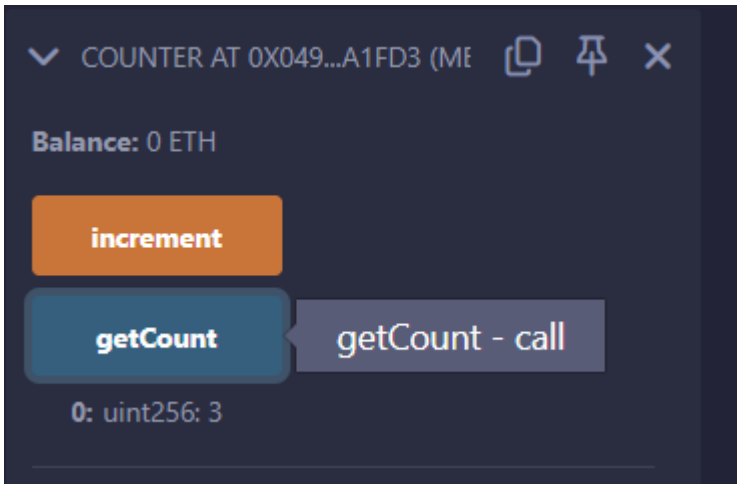
Aim- Implement & demonstrate the use of solidity programming

a) Counter

Code:

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 <0.9.0;
contract counter {
 uint256 private count=0;
 function getCount() public view returns (uint256) {
 return count;
 }
 function increment () public {
 count +=1;
 }
}
```

Output:

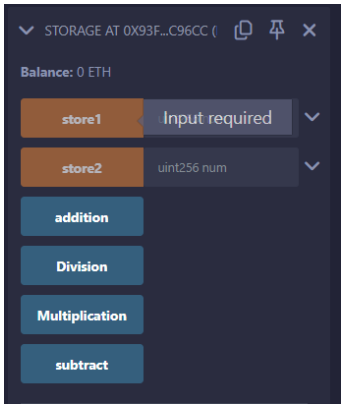


B) Calculator Code:-

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.2 <0.9.0;
/**
* @title Storage
* @dev Store & retrieve value in a variable
* @custom:dev-run-script ./scripts/deploy_with_ethers.ts
*/
contract Storage {
 uint256 number1;
 uint256 number2;
 function store1(uint256 num) public {
 number1 = num;
 }
 function store2(uint256 num) public {
 number2 = num;
 }
 /**
 * @dev Return value
 * @return value of 'number'
 */
 function addition() public view returns (uint256){
 return number1+number2;
 }
 function subtract() public view returns (uint256){
 return number1-number2;
 }
 function Multiplication() public view returns (uint256){
 return number1*number2;
 }
```

```
function Division() public view returns (uint256){
return number1/number2;
}
}
```

Output:



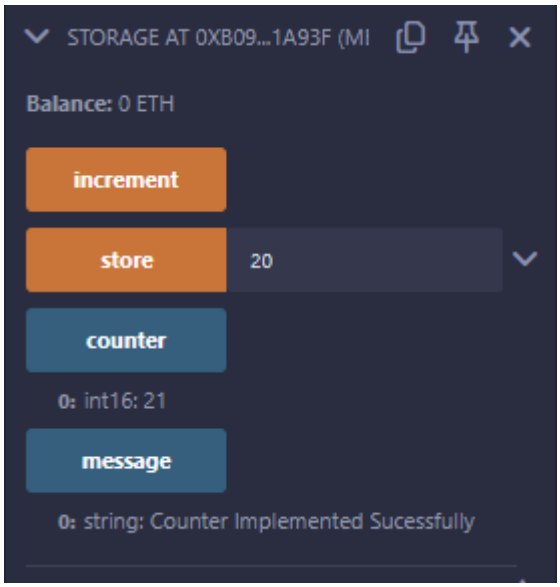
c) Increment operator

Code:

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.8.2 <0.9.0;

/**
 * @title Storage
 * @dev Store & retrieve value in a variable
 * @custom:dev-run-script ./scripts/deploy_with_ethers.ts
 */
contract Storage {

    int16 number;

     function store(int16 num) public {
        number = num;
    }

    function increment() public{
        number++;
    }

    function counter() public view returns (int16){
        return number;
    }
```

```
    function message() public pure returns (string memory) {
        return "Counter Implemented Sucessfully";

    }
}
```

Output:



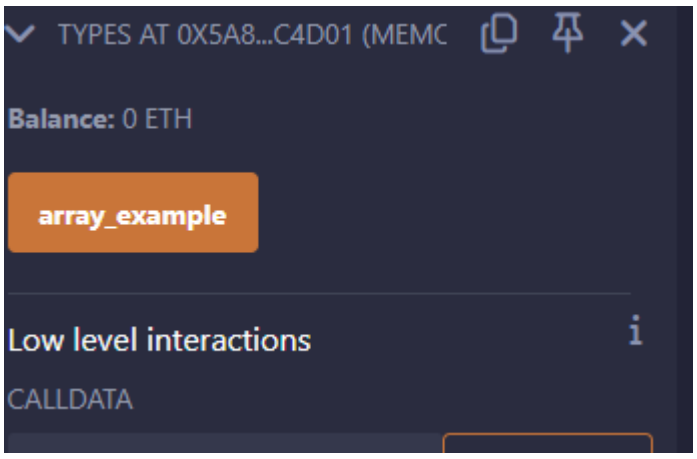
Practical No: 3

Aim: Create a Smart Contract in solidity program to demonstrate array and its types.

Code:

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.6 <0.9.0;

// Creating a contract
contract Types {

    // Declaring state variables
    // of type array
    uint[6] data1;

    // Defining function to add
    // values to an array
    function array_example() public returns (
    int[5] memory, uint[6] memory){

        int[5] memory data
        = [int(50), -63, 77, -28, 90];
        data1
        = [uint(10), 20, 30, 40, 50, 60];

        return (data, data1);
  }
}
```

Output:



```
decoded output                {
                                      "0": "int256[5]: 50,-63,77,-28,90",
                                      "1": "uint256[6]: 10,20,30,40,50,60"
                              }
```

Practical No: 4

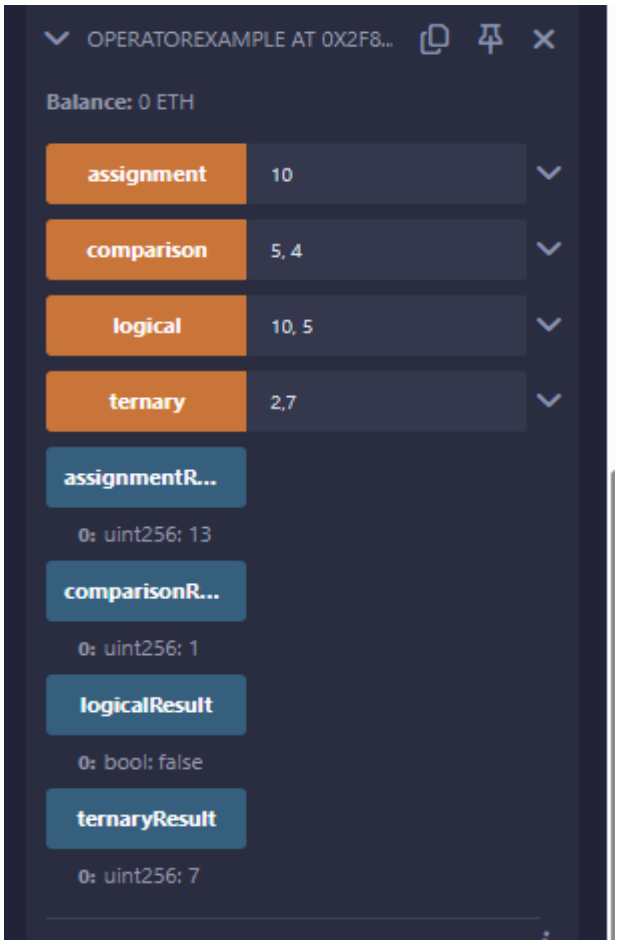
Aim: Create a Smart Contract for Operators in solidity Programming

6. **Solidity program to demonstrate Assignment operators.**
7. **Solidity program to demonstrate Comparison operators.**
8. **Solidity program to demonstrate Logical operators.**
9. **Solidity program to demonstrate Ternary operators.**

Code:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract OperatorExample {
    uint256 public comparisonResult;
    bool public logicalResult;
    uint256 public assignmentResult;
    uint256 public ternaryResult;

    function comparison(uint256 a, uint256 b) public {
        if (a == b) {
            comparisonResult = 0; // Equal
        } else if (a > b) {
            comparisonResult = 1; // Greater Than
        } else {
            comparisonResult = 2; // Less Than
        }
    }

    function logical(bool a, bool b) public {
        logicalResult = (a && b) || (!a && !b); // Logical XOR
    }

    function assignment(uint256 a) public {
        assignmentResult = a;
        assignmentResult += 10; // Addition Assignment
        assignmentResult *= 2;  // Multiplication Assignment
        assignmentResult /= 3;  // Division Assignment
    }

    function ternary(uint256 a, uint256 b) public {
        ternaryResult = (a > b) ? a : b; // Ternary Operator
    }
}
```

Output:


OPERATOREXAMPLE AT 0X2F8...
Balance: 0 ETH
assignment
10
comparison
5, 4
logical
10, 5
ternary
2,7
assignmentR...
0: uint256: 13
comparisonR...
0: uint256: 1
logicalResult
0: bool: false
ternaryResult
0: uint256: 7

Pratical No: 5

Aim: Create a Smart Contract for Loops in Solidity programming

1. **Create a smart contract for loop**
2. **Create a smart contract for while loop**

Code:

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.6.12 <0.9.0;

// Creating a contract
contract Loops {


    //for loop
    uint256 private result =0;
    function forloop(uint256 number)public returns(uint256){
        for(uint256 i=1;i<=number;i++){
            result += i;
        }
        return result;
    }

        function getforloop() public view returns (uint256){
        return result;
    }


     //while loop
    uint256 private result2=0;

    function whileloop(uint256 number)public returns(uint256){
    uint256 i =1;
    while(i <= number){
        result2 +=i;
        i++;
    }
    return result2;

    }

    function getwhileloop() public view returns (uint256){
        return result2;
    }


}
```

Output:


LOOPS AT 0XC9D...F2158 (MEM
Balance: 0 ETH
forloop
5
whileloop
7
getforloop
0: uint256: 15
getwhileloop
0: uint256: 28

Pratical No: 6

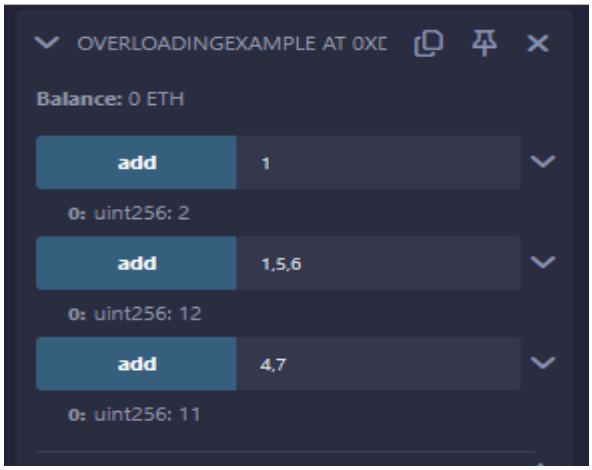
Aim: Perform a Mathematical Function & Function Overloading in solidity Programming.

Code:

1. **Function Overloading**

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract OverloadingExample {

    // Function with one parameter
    function add(uint256 a) public pure returns (uint256) {
        return a + 1;
    }

    // Overloaded function with two parameters
    function add(uint256 a, uint256 b) public pure returns (uint256) {
        return a + b;
    }

    // Another overloaded function with two parameters of different types
    function add(uint256 a, uint256 b, uint256 c) public pure returns
(uint256) {
        return a + b + c;
    }
}
```

Output:

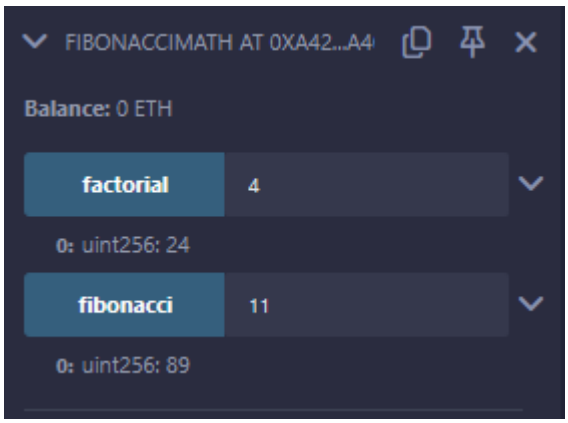


2. Mathematical Function(Fibonacci Sequence and Factorial function)

Code:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract FibonacciMath {

    // Function to calculate the nth Fibonacci number
    function fibonacci(uint256 n) public pure returns (uint256) {
        if (n == 0) return 0;
        if (n == 1) return 1;
        uint256 a = 0;
        uint256 b = 1;
        uint256 c;
        for (uint256 i = 2; i <= n; i++) {
            c = a + b;
            a = b;
            b = c;
        }
        return b;
    }
     // Function to calculate the factorial of a number
    function factorial(uint256 n) public pure returns (uint256) {
        if (n == 0) return 1;
        uint256 result = 1;
        for (uint256 i = 1; i <= n; i++) {
            result *= i;
        }
        return result;
    }
}
```

Output:


FIBONACCIMATH AT 0XA42...A4
Balance: 0 ETH
factorial
4
0: uint256: 24
fibonacci
11
0: uint256: 89

Practical No: 7

Aim: Implementation of Interface & Inheritance in solidity programming.

1. Implementation of Interface

Code:

InterfaceExample.sol

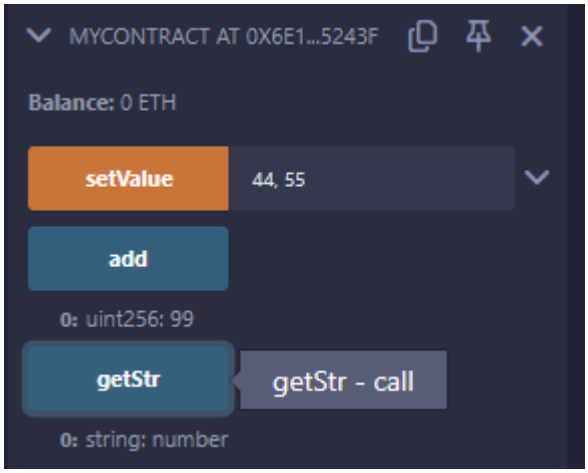
```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.6 <0.9.0;
//initialize the interface
interface InterfaceExample{

    // Functions having only
    // declaration not definition
    function getStr(
    ) external view returns(string memory);

    function setValue(
    uint _num1, uint _num2) external;

    function add(
    ) external view returns(uint);
}
```

Interface_sol.sol

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.6 <0.9.0;
/// @title A contract for demonstrate the working of the interface
/// @author Jitendra Kumar
/// @notice For now, this contract just show how interface implements in the
smart contract

import "./InterfaceExample.sol";
// Contract that implements interface
contract MyContract is InterfaceExample{

    // Private variables
    uint private num1;
    uint private num2;

    // Function definitions of functions
    // declared inside an interface
    function getStr() public view virtual override returns(string memory){
        return "number";
```

```
    }

    // Function to set the values
    // of the private variables
    function setValue(
    uint _num1, uint _num2) public virtual override{
        num1 = _num1;
        num2 = _num2;
    }

    // Function to add 2 numbers
    function add(
    ) public view virtual override returns(uint){
        return num1 + num2;
    }
}

contract call{

    //Creating an object
    InterfaceExample obj;

    constructor(){
        obj = new MyContract();
    }

    // Function to print string
    // value and the sum value
    function getValue(
    ) public returns(string memory,uint){
        obj.setValue(10, 16);
        return (obj.getStr(),obj.add());
    }
}
```

Output:



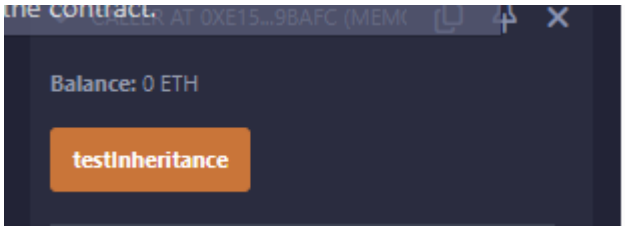
2. Inheritance

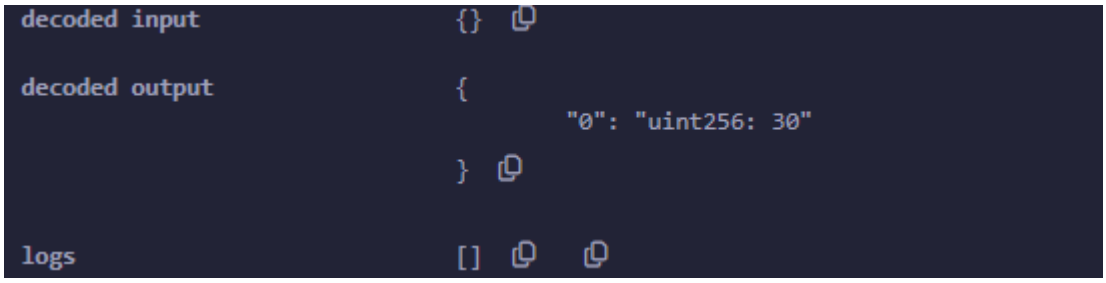
Code:

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.6.12 <0.9.0;



// Solidity program to demonstrate Interface and Inheritance
// Defining an interface
interface IParent {
    function setValue(uint _a, uint _b) external;
    function getValue() external view returns (uint);
}

// Defining a base contract
contract Parent {
    uint internal sum;

    // Function to set the value of sum
    function setValue(uint _a, uint _b) external virtual {
        sum = _a + _b;
    }
}

// Defining a derived contract that inherits from Parent and implements
IParent
contract Child is Parent, IParent {

    // Override function to set the value of sum from both Parent and IParent
    function setValue(uint _a, uint _b) external override(Parent, IParent) {
        sum = _a + _b;
    }
```

```
    // Override function to get the value of sum from IParent
    function getValue() external view override returns (uint) {
        return sum;
    }
}

// Defining a calling contract
contract Caller {

    // Creating Child contract object
    Child cc = new Child();

    // Function to call setValue and getValue functions
    function testInheritance() public returns (uint) {
        cc.setValue(10, 20);
        return cc.getValue();
    }
}
```

Output:





Practical No: 8

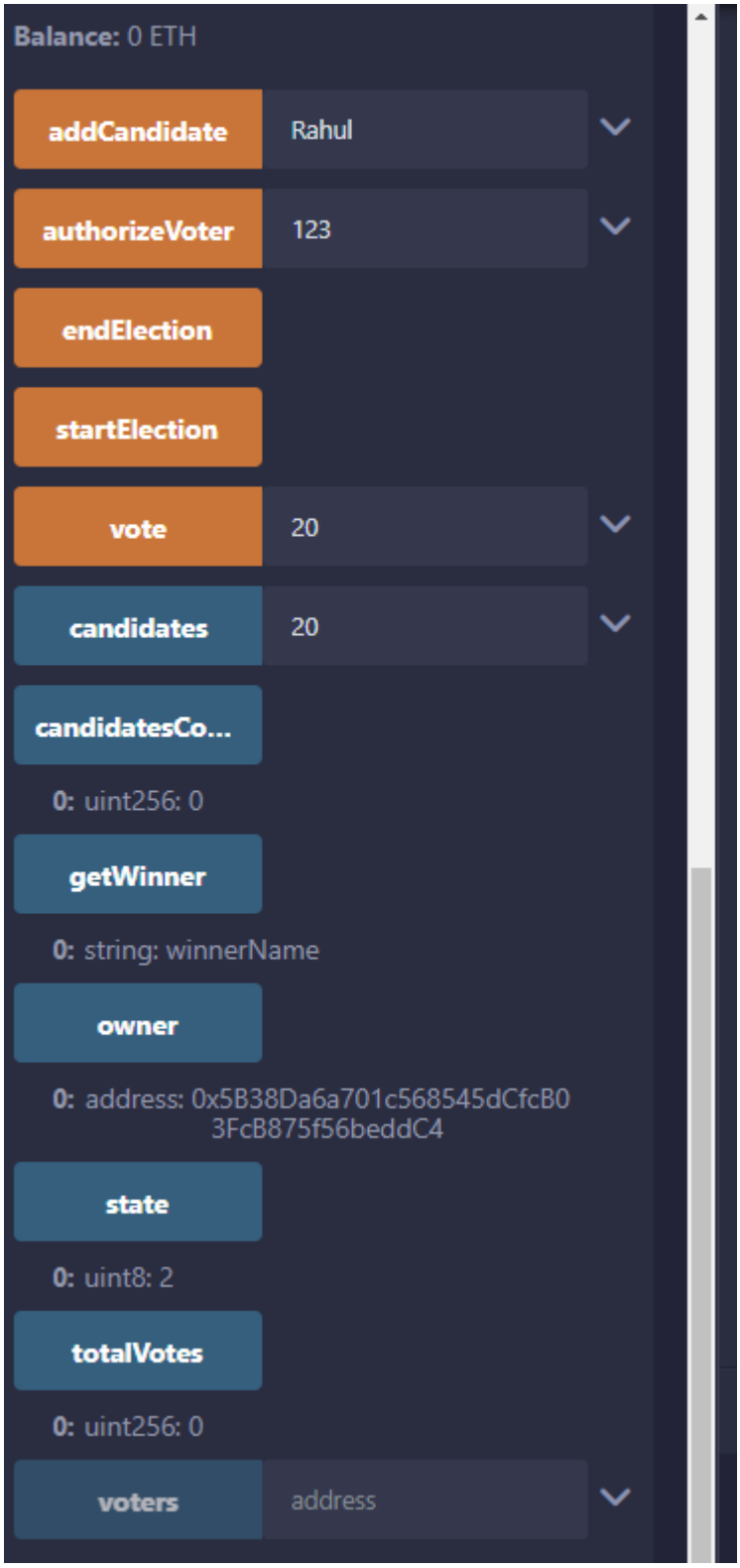
Aim: Create smart contract for Selection of candidate in election.

Code:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Election {
    // Owner of the contract
    address public owner;

    // Election states
    enum ElectionState { Created, Voting, Ended }
    ElectionState public state;

    // Candidate struct
    struct Candidate {
        uint id;
        string name;
        uint voteCount;
    }

    // Voter struct
    struct Voter {
        bool authorized;
        bool voted;
        uint vote;
    }

    // Candidates
    mapping(uint => Candidate) public candidates;
    uint public candidatesCount;

    // Voters
    mapping(address => Voter) public voters;
    uint public totalVotes;

    // Events
    event ElectionStarted();
    event ElectionEnded();
    event VoterAuthorized(address voter);
    event VoteCast(address voter, uint candidateId);

    // Modifiers
    modifier ownerOnly() {
        require(msg.sender == owner, "Only owner can call this function");
        _;
```

```
    }

    modifier inState(ElectionState _state) {
        require(state == _state, "Invalid state for this action");
        _;
    }

    constructor() {
        owner = msg.sender;
        state = ElectionState.Created;
    }

    // Function to add a candidate
    function addCandidate(string memory _name) public ownerOnly
inState(ElectionState.Created) {
        candidatesCount++;
        candidates[candidatesCount] = Candidate(candidatesCount, _name, 0);
    }

    // Function to authorize a voter
    function authorizeVoter(address _voter) public ownerOnly
inState(ElectionState.Created) {
        voters[_voter].authorized = true;
        emit VoterAuthorized(_voter);
    }

    // Function to start the election
    function startElection() public ownerOnly inState(ElectionState.Created) {
        state = ElectionState.Voting;
        emit ElectionStarted();
    }

    // Function to end the election
    function endElection() public ownerOnly inState(ElectionState.Voting) {
        state = ElectionState.Ended;
        emit ElectionEnded();
    }

    // Function to vote
    function vote(uint _candidateId) public inState(ElectionState.Voting) {
        require(voters[msg.sender].authorized, "You are not authorized to
vote");
        require(!voters[msg.sender].voted, "You have already voted");
        require(_candidateId > 0 && _candidateId <= candidatesCount, "Invalid
candidate ID");

        voters[msg.sender].voted = true;
        voters[msg.sender].vote = _candidateId;
```

```
        candidates[_candidateId].voteCount++;
        totalVotes++;

        emit VoteCast(msg.sender, _candidateId);
    }

    // Function to get the winner
    function getWinner() public view inState(ElectionState.Ended) returns
(string memory winnerName) {
        uint maxVotes = 0;
        uint winningCandidateId = 0;

        for (uint i = 1; i <= candidatesCount; i++) {
            if (candidates[i].voteCount > maxVotes) {
                maxVotes = candidates[i].voteCount;
                winningCandidateId = i;
            }
        }

        winnerName = candidates[winningCandidateId].name;
    }
}
```


Balance: 0 ETH
addCandidate
Rahul
authorizeVoter
123
endElection
startElection
vote
20
candidates
20
candidatesCo...
0: uint256: 0
getWinner
0: string: winnerName
owner
0: address: 0x5B38Da6a701c568545dCfcB0
3FcB875f56beddC4
state
0: uint8: 2
totalVotes
0: uint256: 0
voters
address

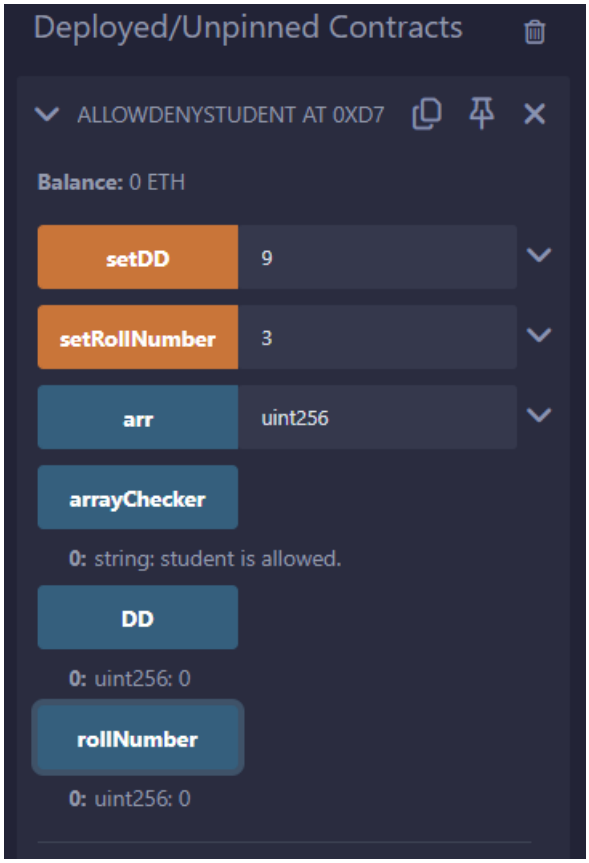
Practical No: 9

Aim: Write a solidity program to create an array of role no.& create a smart contract where it checks the value of roll no.s & perform AND operation with today's date DD and if the result is even display a message " Student is ALLOWED." else display "DENIED".

Code:

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.2;

contract allowDenyStudent {
    uint256[] public arr = [1,2,3,4,5,6,7,8,9,10];
    uint256 public rollNumber;
    uint256 public DD;

    function setRollNumber(uint256 _rollNumber) public {
        rollNumber = arr[_rollNumber];
    }

    function setDD(uint256 _DD) public {
        DD = _DD;
    }

    function arrayChecker() public view returns (string memory) {
        string memory finalOutput ;

        uint256 operation = rollNumber & DD;

        if (operation % 2 == 0) {
            finalOutput = 'student is allowed.';

        }else {
            finalOutput = 'student is denied.';
        }

        return finalOutput;
    }


}
```

Output:


Deployed/Unpinned Contracts
ALLOWDENYSTUDENT AT 0XD7
Balance: 0 ETH
setDD
9
setRollNumber
3
arr
uint256
arrayChecker
0: string: student is allowed.
DD
0: uint256: 0
rollNumber
0: uint256: 0

Practical No: 10

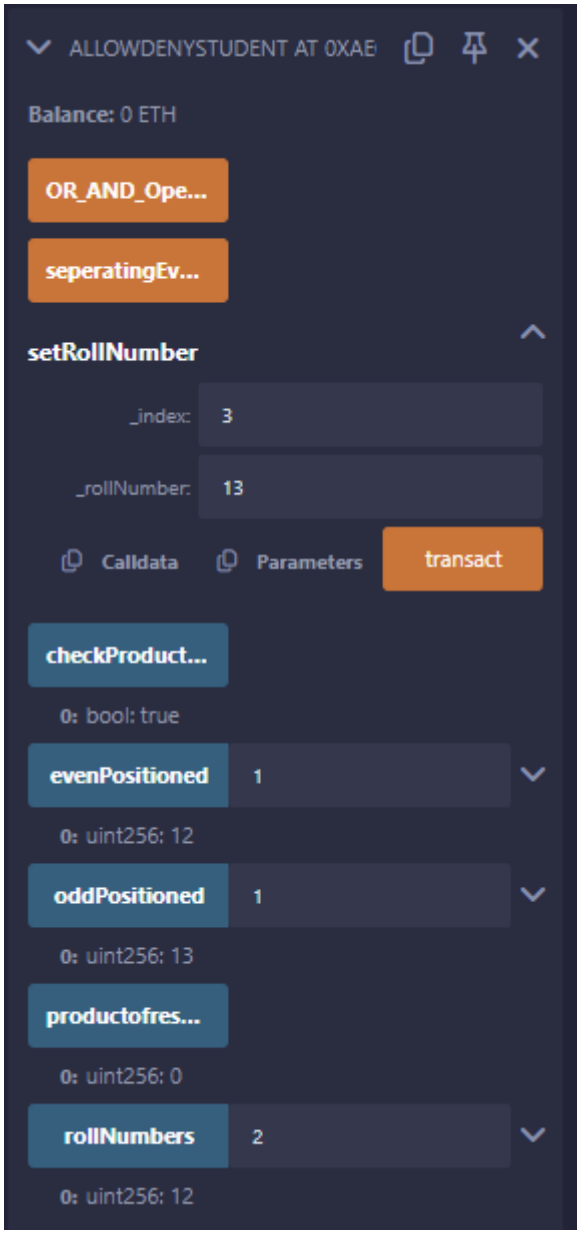
Aim: - Write a solidity program to find the sum of an array of ten numbers using loop the numbers are expected to be taken from the user, create a smart contract to find the AND operation of odd positioned numbers and OR operation of even positioned numbers including 0^{th} index. Hence find the product of the results and also identify whether the result is the part of array or not.

Code:

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.6 <0.9.0;
import "hardhat/console.sol";
contract allowDenyStudent {
    uint256[10] public rollNumbers;
    function setRollNumber (uint _index,uint _rollNumber) public {
        rollNumbers[_index] = (_rollNumber);
    }
    function seperatingEvenOdd() public {
    for (uint i=0; i<rollNumbers.length; i++) {
        if(i %2 == 0) {
            evenPositioned.push(rollNumbers[i]);
        } else {
            oddPositioned.push(rollNumbers[i]);
        }}}
    uint[] public evenPositioned;
    uint[] public oddPositioned;
    function OR_AND_Operation() public {
        uint resultOfOR = evenPositioned[0];
        for (uint i=1;i<evenPositioned.length; i++) {
            resultOfOR = resultOfOR | evenPositioned[i];
        }
        uint resultofAND = oddPositioned[0];
        for (uint i=1;i<oddPositioned.length; i++) {
            resultofAND = resultofAND & oddPositioned[i];
        }
        productofresults = resultOfOR = resultofAND;
    }
    uint256 public productofresults;
    function checkProductOfResults() public view returns (bool) {
    bool result;
    for (uint i=0; i<rollNumbers.length; i++) {
            result = (productofresults == rollNumbers [i]) ? true : false;
        }
        return result;
```

```
    }
}
```

Output:


ALLOWDENYSTUDENT AT 0XAE
Balance: 0 ETH
OR_AND_Ope...
seperatingEv...
setRollNumber
_index: 3
_rollNumber: 13
Calldata
Parameters
transact
checkProduct...
0: bool: true
evenPositioned
1
0: uint256: 12
oddPositioned
1
0: uint256: 13
productofres...
0: uint256: 0
rollNumbers
2
0: uint256: 12