INDEX

Sr.No	Topic		Date	Remark
1	**Write the following programs for Blockchain in Python :** i. A Simple client class that generates the private and public keys by using the built in Python RSA algorithm and test it. ii. A transaction class to send and receive money and test it.			
2	**Write the following programs for Blockchain in Python :** i. Create multiple transactions and display them. ii. Create a blockchain, a genesis block and execute it.			
3	**Write the following programs for Blockchain in Python :** i. Create a mining function and test it. ii. Add blocks to the miner and dump the blockchain.			
4	**Implement and demonstrate the user of the following in Solidity :** i. Variable ii. Operations iii. Loops iv. Decision Making v. Strings			
5	**Implement and demonstrate the user of the following in Solidity :** i. Arrays ii. Enums iii. Structs iv. Mappings v. Conversations vi. Ether Units vii. Special Variables			

6	**Implement and demonstrate the user of the following in Solidity:** i. Functions ii. View Functions iii. Pure Functions iv. Fallback Functions v. Function Overloading vi. Mathematical Functions vii. Cryptographic Functions			
7	**Implement and demonstrate the user of the following in Solidity:** i. Contracts ii. Inheritance iii. Constructors iv. Abstract Class v. Interfaces			
8	**Implement and demonstrate the user of the following in Solidity:** i. Libraries ii. Assembly iii. Events iv. Error Handling			

PRACTICAL 1

1.
2.
3.
4.
5.
6.
7.
8.
9.

PRACTICAL 2

```
========================================================== RESTART: C:/Users/arsha/blockchain/prac2a.py ========================================
Transaction:  1
sender:30819f300d06092a864886f70d010101050003818d003081890281810 0ddac439cdb0b3c2326959cff808660c609e5c025692b8819a15488eafc9802f3d5c8fbe4c802628
1eee4b7512ef1c3d35b7b3af2455fd1e397299db278ed75187d21f908c4f71dea9726c76f7119fc5ebb69fa210203010001
-----
recipent:30819f300d06092a864886f70d010101050003818d0030818902818100b0a3621386b18c2e6132fdf442153e12da7b7d55ba5cf893851d86a0029798228487367cf4c13
a9238b11e601d62ed8a7b15fb99afc820e9e0fd0fdd575d2eed15b4550f4293bb383ab2594a53e7e82f6937a5d30203010001
-----
value:15.0
-----
time:2023-04-20 02:56:29.665957
-----
-------------------
Transaction:  2
sender:30819f300d06092a864886f70d010101050003818d0030818902818100b0a3621386b18c2e6132fdf442153e12da7b7d55ba5cf893851d86a0029798228487367cf4c1302
238b11e601d62ed8a7b15fb99afc820e9e0fd0fdd575d2eed15b4550f4293bb383ab2594a53e7e82f6937a5d30203010001
-----
recipent:30819f300d06092a864886f70d010101050003818d0030818902818100bef9927c52d3bfa9ad83eb04edfbb3b8e38847be775a91370d05285dc106a1e45a850daa1a963
5d41583b3d01e6b359d86c18e2c0a52303ad8c3ff480f8f60513fa9150a0791be06018a67f53aef7f80e26635ed0203010001
-----
value:17.0
-----
time:2023-04-20 02:56:29.668956
-----
-------------------
Transaction:  3
sender:30819f300d06092a864886f70d010101050003818d0030818902818100bef9927c52d3bfa9ad83eb04edfbb3b8e38847be775a91370d05285dc106a1e45a850daa1a963a8
41583b3d01e6b359d86c18e2c0a52303ad8c3ff480f8f60513fa9150a0791be06018a67f53aef7f80e26635ed0203010001
-----
recipent:30819f300d06092a864886f70d010101050003818d0030818902818100b0a3621386b18c2e6132fdf442153e12da7b7d55ba5cf893851d86a0029798228487367cf4c13
a9238b11e601d62ed8a7b15fb99afc820e9e0fd0fdd575d2eed15b4550f4293bb383ab2594a53e7e82f6937a5d30203010001
-----
value:10.0
-----
time:2023-04-20 02:56:29.670970
-----
```

Output:

```
>
========================================================================================== RESTART: C:/Users/arsha/blockchain/prac2b.py ==============
Number of blocks in the chain:1
block#0
sender:Genesis
-----
recipent:30819f300d06092a864886f70d010101050003818d003081890281810 09f096a507f216802bb29b8a1c37a47d9c06f4ab60018d48791bd25bd6b0ac9e891ba74e5aa65122fe716fb
6a431ecdf7ca1b477d1778c2d6ac1693d65acb54d06de78aaca9fb9c3a84ec5f7f883416f33954316239cb8b3090203010001
-----
value:500.0
-----
time:2023-04-20 03:06:29.650204
-----
''''''''''''''
==================
>
```

Output:

```
>
==========================================================================================
========= RESTART: C:/Users/arsha/blockchain/prac3a.py ==============================
==================================================================
Rifath,3
prefix 111
Testing --> 9663f9ef9867d30958fbe492bc67fe33d7f13637628884f35b0b7387326a1f06
Testing --> 2af26dd05395e19b5761463f09f22b1a8063887361d29c406bb1ac1c877d4755
Testing --> 76e35ecb0e034a434f2eef033ba5192f96e1a40fcf97e70560cdfcd4de5d8ab2
Testing --> c9abcb71c9cdac4797d2ba0c710c2598a695d70bcc3cf7309e1d976ac7ac468c
Testing --> 2952367650291a62f86409b92baebb9e598bfa29eaf558716d13259bbc21eddf
Testing --> 59a8f70aee0759c4b752cfbb6f3e12645ef778c71fae802b648a92a2d9238b42
Testing --> d2ded8fbddf21150c31b2cf5f9c2c5144968654801bd53af4c8d50b371f222d0
Testing --> 8de197364275d2701eac232c76a84d4cd1800c7154bda8adcafc594742e03a59
Testing --> 1f7ca432198f140880913cfd04a744e33be50de7152bb08e88cfd970e907da1c
Testing --> c8f3b6d29aa95dc7c63cff82be0ce8f3a4a648c00e1c78812ee8bb2af4db93b3
Testing --> 0254f4722bb5fff567cb5e9228d9d89c26a6ac743e42b7dd567d2dafd6121d8e
Testing --> deff7bec76f33d782a150da55e9db1aaf44a28dcfbce67ad4e892bc85a2ffd8d
Testing --> e09ddd5493a45835abc3d0329e33ea86cdc4cfe5b4cd0d725b8fa114cfdbe409
Testing --> 3b08cf5e99a287be78d24a616bea5d0fe2d267c7cc28531196989de6527251cb
Testing --> 41404c1f666b674012d942493db6d26f24ac3e04082692c6ee89e04b6833d0e8
Testing --> a56cf3972dd497f9b7cc96cab601cbdc957c5178c92fdf4802c057727c627d1c
```

Output:

```
========================================================================================== RESTART: C:/Users/arsha/blockchain/prac3a.py ===
==================================================================
Rifath,3
prefix 111
Testing --> 9663f9ef9867d30958fbe492bc67fe33d7f13637628884f35b0b7387326a1f06
Testing --> 2af26dd05395e19b5761463f09f22b1a8063887361d29c406bb1ac1c877d4755
Testing --> 76e35ecb0e034a434f2eef033ba5192f96e1a40fcf97e70560cdfcd4de5d8ab2
Testing --> c9abcb71c9cdac4797d2ba0c710c2598a695d70bcc3cf7309e1d976ac7ac468c
Testing --> 2952367650291a62f86409b92baebb9e598bfa29eaf558716d13259bbc21eddf
Testing --> 59a8f70aee0759c4b752cfbb6f3e12645ef778c71fae802b648a92a2d9238b42
Testing --> d2ded8fbddf21150c31b2cf5f9c2c5144968654801bd53af4c8d50b371f222d0
Testing --> 8de197364275d2701eac232c76a84d4cd1800c7154bda8adcafc594742e03a59
```

```
Testing --> 1168a0625dc5949075a6bb2abbbb39c7b216282c6b390b840eca32ea4ec32c75
After 56iterations found nounce 1168a0625dc5949075a6bb2abbbb39c7b216282c6b390b840eca32ea4ec32c75
prefix 11
Testing --> 9e3361080864588fd58a109d15c8062bf723118f51cb86f88c9ee06a135eea54
Testing --> 9b32d74d4612b3cad4bd2c97ee6f65f5103c632934b7ce24bf6c78f07a227c22
Testing --> ba25be857b1034fe457620845576aa0d16fef898b7a12fb83d28f168bdc54f24
Testing --> 19e95eda815c2001c01c1efb7c639f777245174153a4f13df08168d7aea01c09
```

```
After 56iterations found nounce 1168a0625dc5949075a6bb2abbbb39c7b216282c6b390b840eca32ea4ec32c75
Number of blocks in the chain:3
block#0
sender:Genesis
-----
recipent:30819f300d06092a864886f70d010101050003818d0030818902818100b24ac04cb8a3826afb210b79f5283584cacab09a456ed74fcfd478855d9
68425b1907708a6da74e9dc8aa5df68ba42c03fd4cda0003a546b4212e015407afecee6cefcd666070d339cd1ade1137753b0db6e77234ec618edceceee27f
203010001
-----
value:500.0
-----
time:2023-04-20 03:56:23.648588
-----
...............
====================
block#1
sender:30819f300d06092a864886f70d010101050003818d0030818902818100b24ac04cb8a3826afb210b79f5283584cacab09a456ed74fcfd478855d993
425b1907708a6da74e9dc8aa5df68ba42c03fd4cda0003a546b4212e015407afecee6cefcd666070d339cd1ade1137753b0db6e77234ec618edceceee27f86
3010001
-----
recipent:30819f300d06092a864886f70d010101050003818d0030818902818100b7e289f81620cdd297ac96a77aff42c62d311fb5f4d92126b1bb0aab4f8
7c6bcfc9229de86d0b3e6c3fa133b9505cb32e899385a05bb711b863f3dbadee9df036741c923189f8089822b20ca672c97007bb2d066529e0684a7f85d350
203010001
-----
value:15.0
-----
time:2023-04-20 03:56:23.649589
-----
...............
--------------------
```

```
...............
====================
sender:30819f300d06092a864886f70d010101050003818d0030818902818100b7e289f81620cdd297ac96a77aff42c62d31
6bcfc9229de86d0b3e6c3fa133b9505cb32e899385a05bb711b863f3dbadee9df036741c923189f8089822b20ca672c97007b
3010001
-----
recipent:30819f300d06092a864886f70d010101050003818d0030818902818100bcc8cc45d9e253ec198971e31d918d949f
5395ffec260ffa16f086b35e107f47ce4bbf2ca664dbafee8c3a218c63a27526ea2a87a0571f1e4fb5759170f6a8ec3254309
203010001
-----
value:17.0
-----
time:2023-04-20 03:56:23.652576
-----
...............
====================
block#2
sender:30819f300d06092a864886f70d010101050003818d0030818902818100bcc8cc45d9e253ec198971e31d918d949fa2
95ffec260ffa16f086b35e107f47ce4bbf2ca664dbafee8c3a218c63a27526ea2a87a0571f1e4fb5759170f6a8ec3254309ec
3010001
-----
recipent:30819f300d06092a864886f70d010101050003818d0030818902818100b7e289f81620cdd297ac96a77aff42c62d
7c6bcfc9229de86d0b3e6c3fa133b9505cb32e899385a05bb711b863f3dbadee9df036741c923189f8089822b20ca672c9700
203010001
-----
value:10.0
-----
time:2023-04-20 03:56:23.654577
-----
```

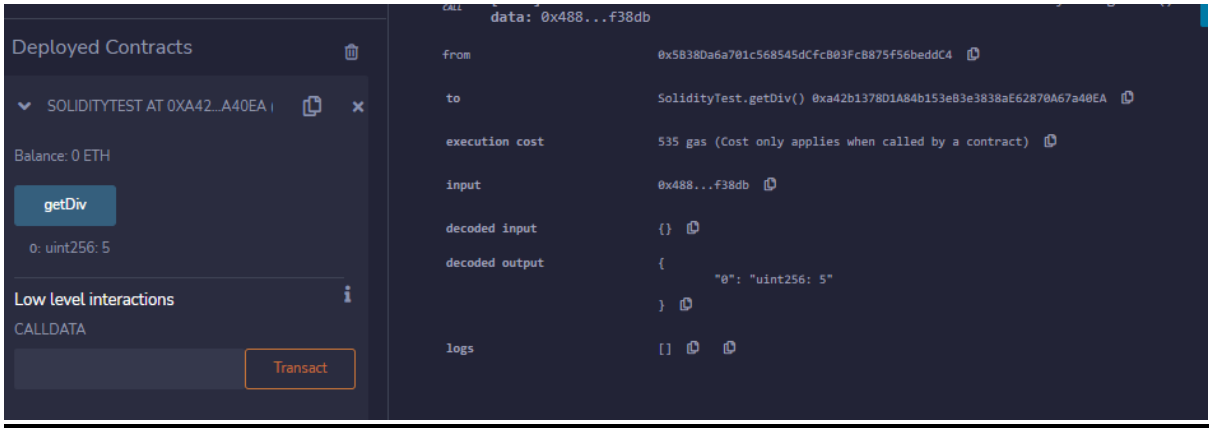
PRACTICAL 4

1. Variable

Code:

```
pragma solidity ^0.8.0;
//RIFATH 3
contract SolidityTest {
   uint storedData; // State variable
   constructor() public{
      storedData=10;
   }
   function getDiv() public view returns(uint){
      uint a=10; // local variable
      uint b=2;
      uint result = a / b;
   return result; // accesss the state variable
}}
```

Output:

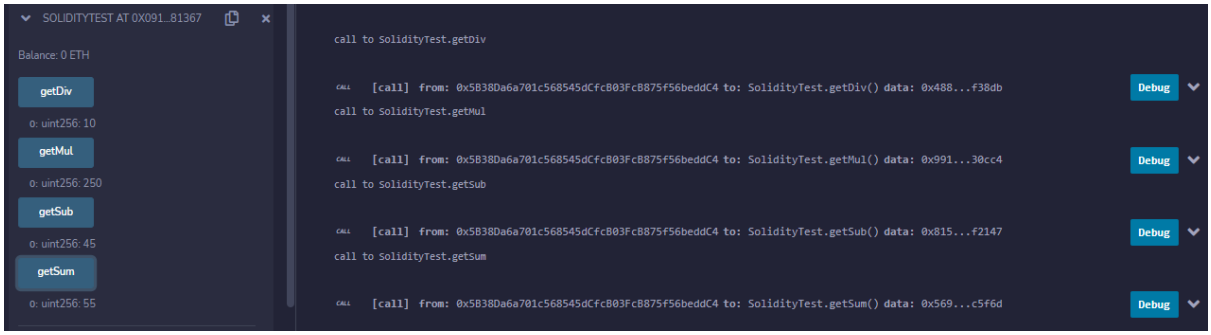


2. Operations

Code:

```
pragma solidity ^0.8.0;
//RIFATH 3
contract SolidityTest {
   uint storedData; // State variable
   constructor() public{
      storedData=10;
   }
   function getDiv() public view returns(uint){
      uint a=50; // local variable
      uint b=5;
      uint result = a / b;
   return result; // accesss the state variable
   }
   function getMul() public view returns(uint){
      uint a=50; // local variable
      uint b=5;
      uint result = a * b;
   return result; // accesss the state variable
   }
   function getSum() public view returns(uint){
      uint a=50; // local variable
      uint b=5;
      uint result = a + b;
   return result; // accesss the state variable
   }
   function getSub() public view returns(uint){
      uint a=50; // local variable
      uint b=5;
      uint result = a - b;
   return result; // accesss the state variable
   }}
```

Output:



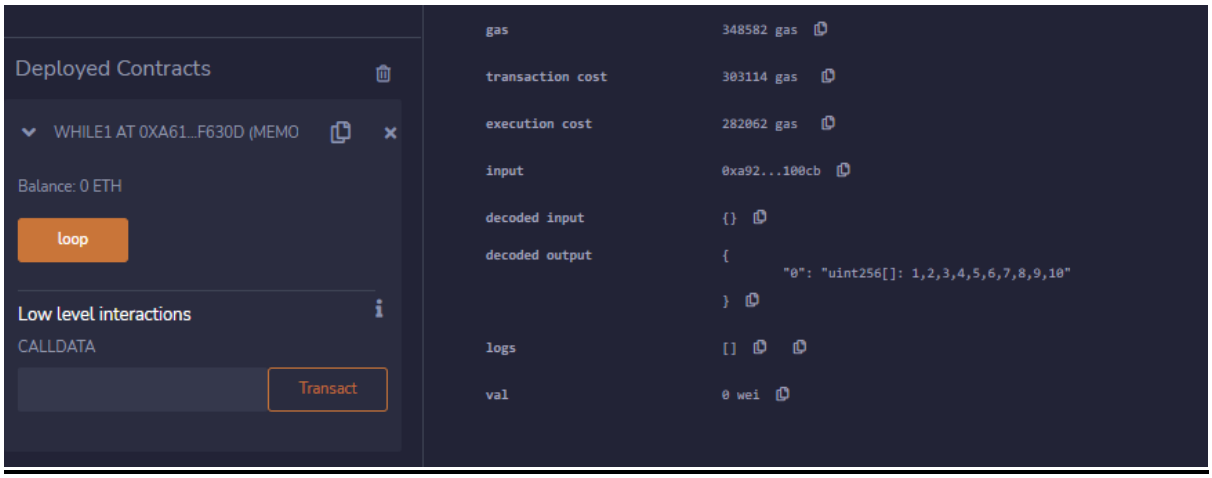
3. Loops

a. While

Code:

```
pragma solidity ^0.8.0;
//rifath 3
contract while1{
uint[] data;
uint8 j=0;
function loop() public returns(uint[] memory)
{
   while (j<10)
   {
      j++;
      data.push(j);
   }
   return data;
}
}
}
```

Output:



Do While

Code:

```
pragma solidity ^0.8.0;
//rifath 3
contract doWhile1{
uint[] data;
uint8 j=0;
function loop() public returns(uint[] memory)
{
   do
   {
      j++;
      data.push(j);
   }
   while (j<10);
   return data;
}
}
```

b. **For**

Code:

```
pragma solidity ^0.8.0;
contract ForLoop{
   function count() public pure returns(uint256){
      uint256 sum=0;
      for(uint256 i=0;i<=25;i++){
         sum+=i;
      }
      return sum;
  }}
```

Output:

4. Decision Making

a. If else

Code:

```
pragma solidity ^0.8.0;
contract Check{
   uint i=100;
   uint j=80;
   function ifElse() public returns(string memory)
   {
      if(i<j)
      {
         return "i is smaller than j";
      }
      else
            {
         return " i is greater than j";
      }  }}
```

Output:

b. If else-if

Code:

```
pragma solidity ^0.8.0;
contract Check{
   uint i=100;
   uint j=100;
   function ifElseIf() public returns(string memory)
   {
      if(i<j)
      {
         return "i is smaller than j";}
      else if(i>j)
      {
         return " i is greater than j"; }
      else
      {
         return " i is equal to j";
     }}}
```

Output:

5. Strings

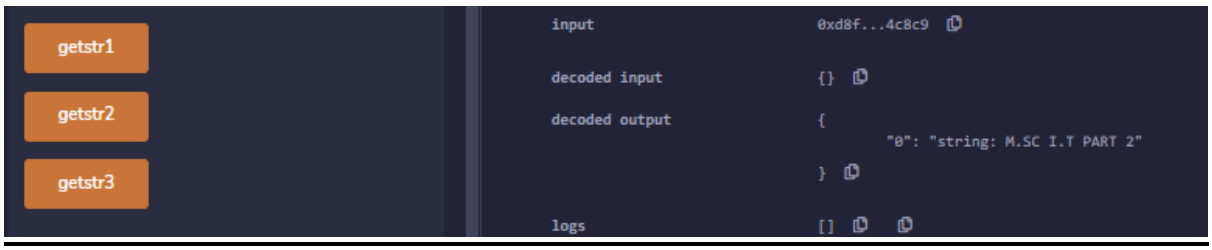
a. Regular string

Code:

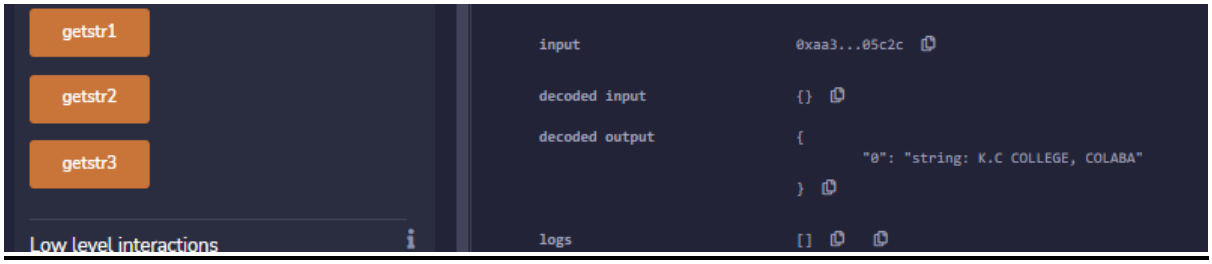
```
pragma solidity ^0.8.0;
contract SS{
   string str1="M.SC I.T PART 2";
   string str2='K.C COLLEGE, COLABA';
   string str3=new string(20);
   function getstr1() public returns(string memory)
   {
      return str1;   }
   function getstr2() public returns(string memory)
   {
      return str2;  }
   function getstr3() public returns(string memory)
   {
      return str3;
   }}
```

Output:

String1:



String2:



String3:

b. Concatenate

Code:

```
pragma solidity >=0.5.0 <0.9.0;
//rifath 3
  contract Demo{
    string public s1 = "RIFATH ";
    string public s2 = "ZAHRAA";
    string public new_str;

    function concatenate() public {
      new_str = string(abi.encodePacked(s1, s2));
     }
}
```

Output:

c. Compare

Code:

```
pragma solidity ^0.8.0;
contract Demo{
   string str1="rifath";
   string str2='rifath”';
   bool public isEqual;
   function cmp() public
   {
      isEqual=keccak256(abi.encodePacked(str1))==keccak256(abi.encodePacked(str2));
   }
}
```

Output:

PRACTICAL 5

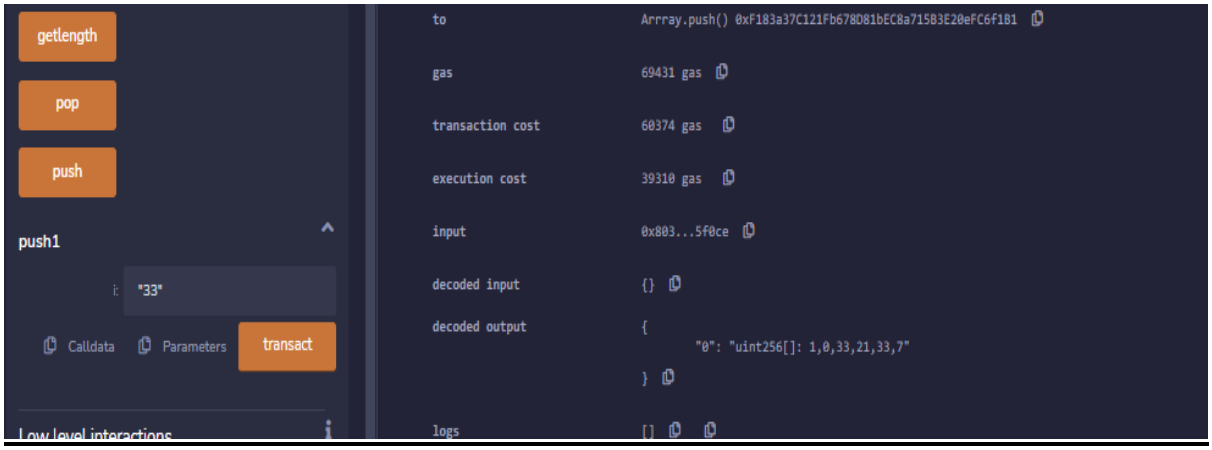
1. Arrays

Code:

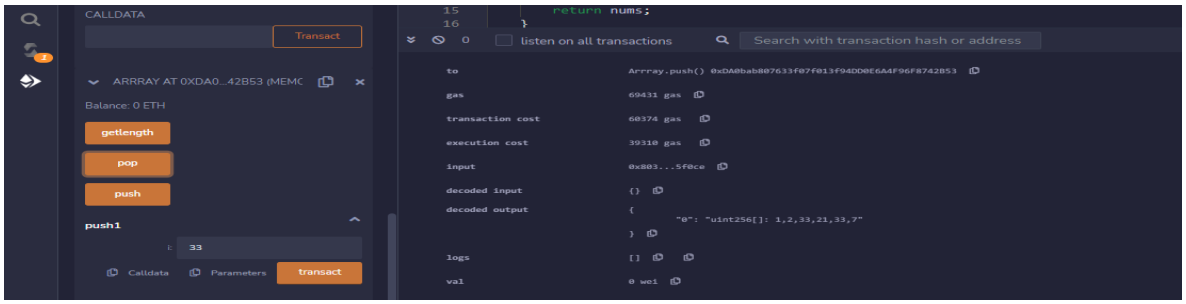
```
pragma solidity ^0.5.0;
contract Arrray{
   uint[] nums=[1,2,33,21];

   function getlength() public returns(uint){
      return nums.length;
   }

   function pop() public{
      delete nums[1];
   }
   function push() public returns (uint[] memory){
      nums.push(7);
      return nums;
   }
function push1(uint i) public{
         nums.push(i);
      }
}
```

Output:

Push:



Pop:



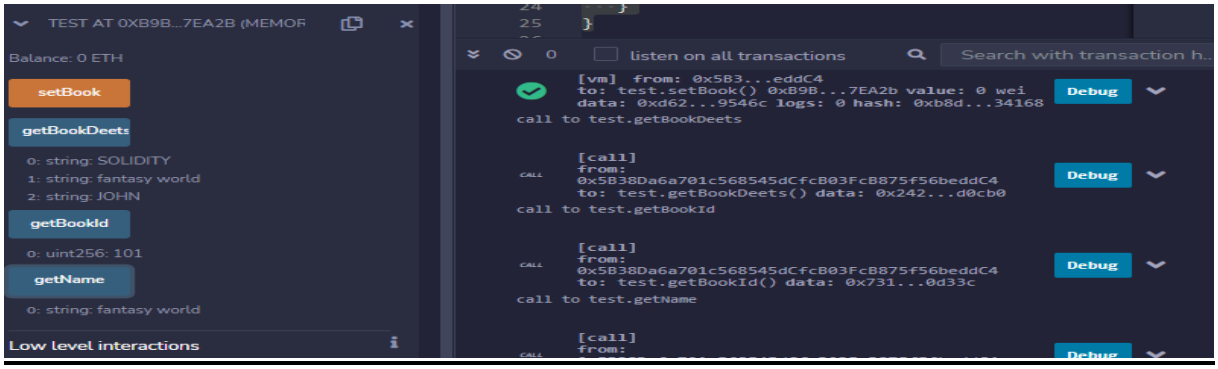
Dynamic:

2. Struct

Code:

```
pragma solidity ^0.5.0;
contract test{
   struct Book{
      string title;
      string author;
      string name;
      uint book_id;
   }
  Book;
  function setBook() public{
     book = Book('SOLIDITY','JOHN','fantasy world',101);
  }

  function getBookId() public view returns(uint){
      return book.book_id;
  }

   function getName() public view returns(string memory){
      return book.name;
  }

  function getBookDeets() public view returns(string memory,string memory){
     return(book.title,book.name,book.author);
     }
}
```

Output:

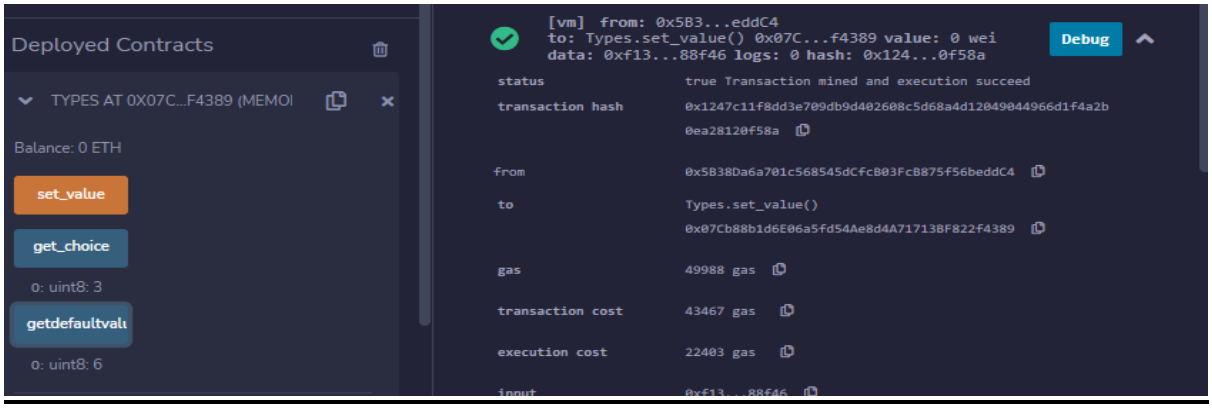


3. Enum

Code:

```
pragma solidity ^0.5.0;

contract Types{

    enum week_days
    {
      Monday,
      Tuesday,
      Wednesday,
      Thursday,
      Friday,
      Saturday,
      Sunday
    }
    week_days week;

    week_days choice;

    week_days constant default_value = week_days.Sunday;

    function set_value() public{
      choice = week_days.Thursday;
    }

    function get_choice() public view returns(week_days){
      return choice;
    }
    function getdefaultvalue() public pure returns(week_days){
      return default_value;
    }
}
```

Output:

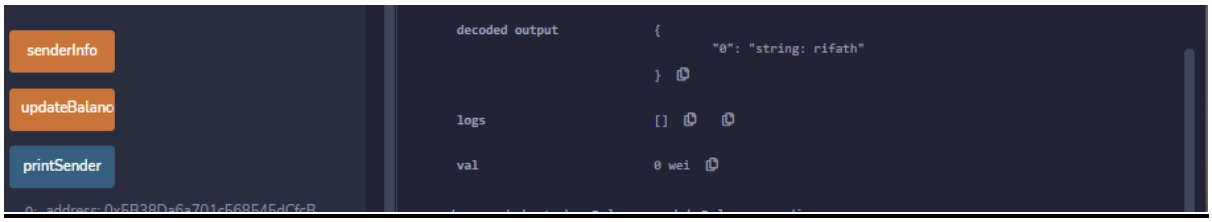


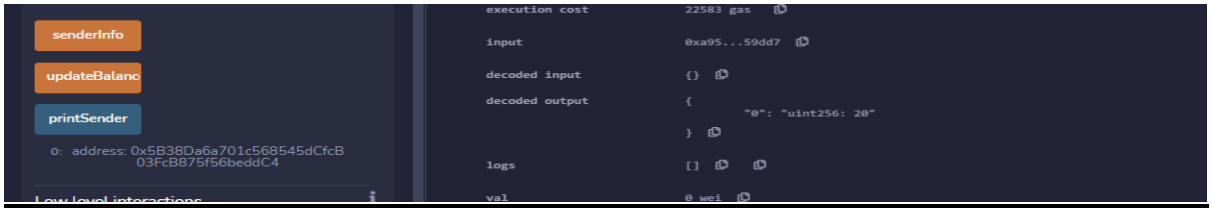
4. Mapping

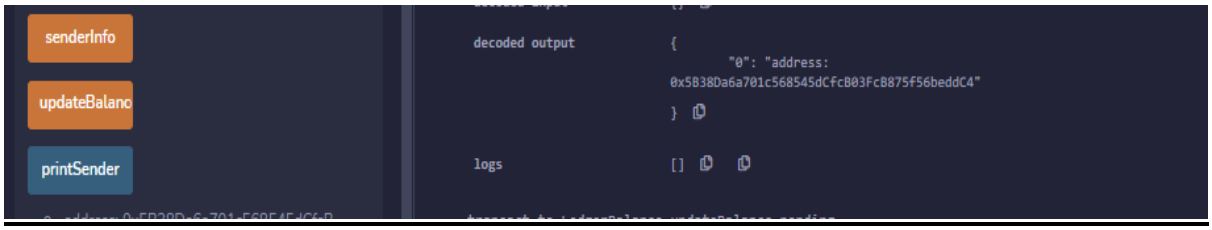
Code:

```
pragma solidity ^0.5.0;
contract LedgerBalance{
   mapping(address => uint) balance;
   mapping(address => string) name;
   function updateBalance() public returns(uint){
         balance[msg.sender]=20;
         return balance[msg.sender];
   }

   function senderInfo() public returns(string memory){
      name[msg.sender] = "rifath";
      return name[msg.sender];
   }
   function printSender() public view returns(address){
      return msg.sender;
   }
}
```

Output:







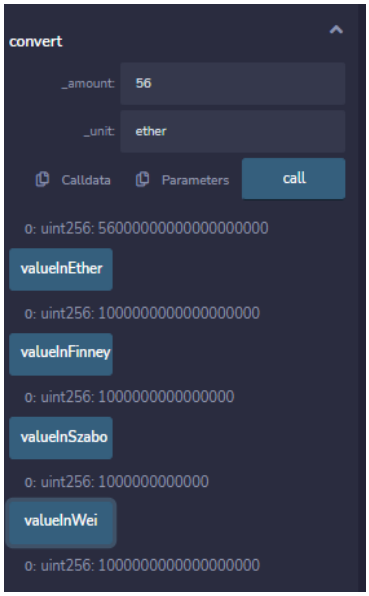
6. Conversions and Ether units

keccak256

Code:

```
pragma solidity >=0.4.0 <0.7.0;
contract EtherUnitsExample {
   uint256 public valueInWei = 1 ether; // 1 ether in Wei
   uint256 public valueInFinney = 1 finney; // 1 finney in Wei
   uint256 public valueInSzabo = 1 szabo; // 1 szabo in Wei
   uint256 public valueInEther = 1 ether; // 1 ether in Wei

   function convert(uint256 _amount, string memory _unit) public pure returns (uint256) {
      if (keccak256(abi.encodePacked(_unit)) == keccak256(abi.encodePacked("wei"))) {
         return _amount;
      } else if (keccak256(abi.encodePacked(_unit)) == keccak256(abi.encodePacked("finney")))
{
         return _amount * 1 finney;
      } else if (keccak256(abi.encodePacked(_unit)) == keccak256(abi.encodePacked("szabo")))
{
         return _amount * 1 szabo;
      } else if (keccak256(abi.encodePacked(_unit)) == keccak256(abi.encodePacked("ether"))
|| keccak256(abi.encodePacked(_unit)) == keccak256(abi.encodePacked("eth"))) {
         return _amount * 1 ether;
      } else {
         revert("Invalid unit");
      }
   }}
```

Output:



7. Special variables:

a. **Solidity contract to demonstrate the special variables block.number and blockhash.**

Code:

```
pragma solidity ^0.5.0;
contract prac
{
   uint BNumber;
   bytes32 BHashPresent;
   bytes32 BHashPrevious;
   function PresentHash()
         public returns(bytes32)
   {
      BNumber = block.number;
      BHashPresent =blockhash(BNumber);
      return BHashPresent;
   }
   function PreviousHash()
         public returns(bytes32)
   {
      BNumber = block.number;
      BHashPrevious = blockhash(BNumber - 1);
      return BHashPrevious;
   }
}
```

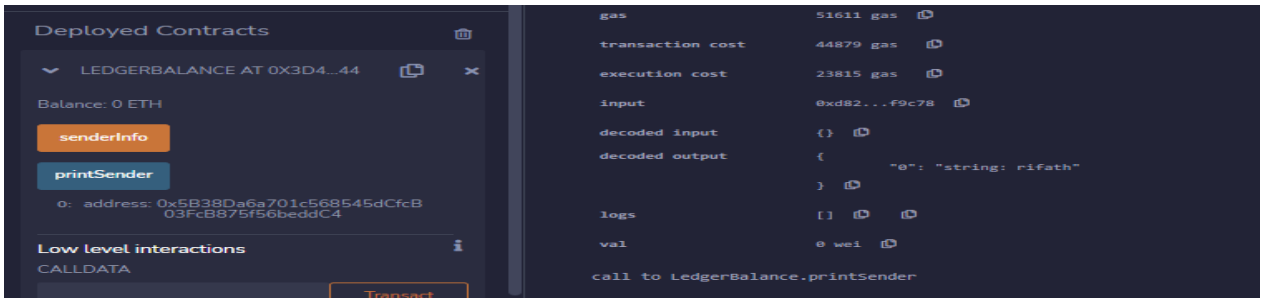
Output:

b. **Solidity contract to demonstrate msg.sender**

Code:

```
pragma solidity ^0.5.0;
contract LedgerBalance{
   mapping(address => string) name;
   function senderInfo() public returns(string memory){
      name[msg.sender] = "rifath";
      return name[msg.sender];
   }

   function printSender() public view returns(address){
      return msg.sender;
   }
}
```

Output:



PRACTICAL 6

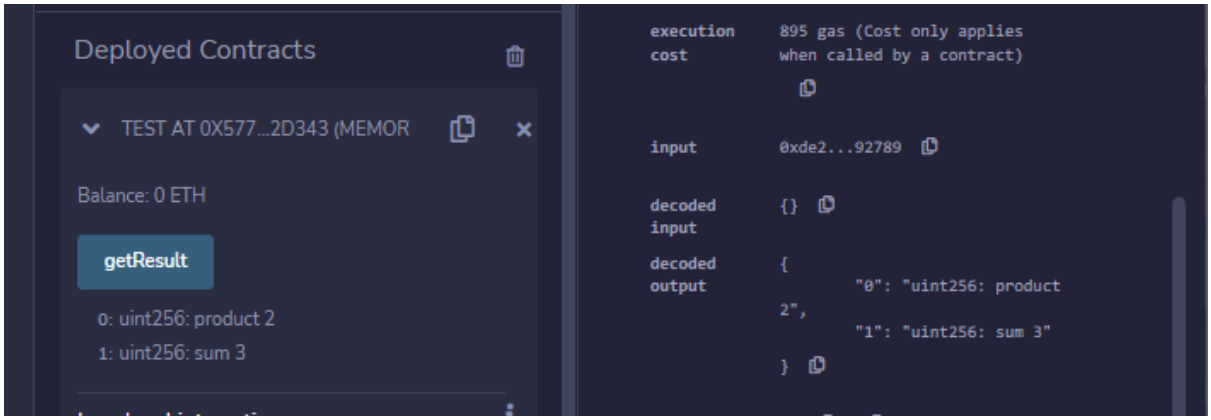
Implement and demonstrate the use of the following in Solidity

1. **Functions**

Code:

```
pragma solidity ^0.8.0;
contract LedgerBalancee {
   mapping(address => string) name;
   function updateBalance() public returns(string memory) {
      name[msg.sender]="zah";
      return name [msg.sender];
   }
   function printsender() public view returns(address){
      return msg.sender;
   }
}
contract Test {
  function getResult() public view returns(uint product, uint sum){
    uint a = 1; // local variable
    uint b = 2;
    product = a * b;
    sum = a + b;
    //return(a*b, a+b);
  }}
```

Output:



2. .View Functions

Code:

```
pragma solidity ^0.5.0;
contract Test{
   function getResult() public view  returns(uint product, uint sum){
      uint a=1;// local variable
      uint b=2;
      product=a*b;
      sum = a+b;
   }
}
```

Output:

3. .Pure Functions

Code:

```
pragma solidity ^0.5.0;
contract Test{
   function getResult() public pure returns(uint product,uint sum){
      uint a = 1; //local variable
      uint b = 2;
      product = a*b;
      sum = a+b;
   }
}
```

Output:

4. .Fallback Functions

Code:

```
pragma solidity ^0.5.12;

contract A {
   uint n;
   function set(uint value) external {
      n=value;
   }

   //fallback function
   function() external payable{
      n=0;
   }
}
contract example{
   function callA(A a) public returns (bool){

      (bool success,) = address(a).call(abi.encodeWithSignature("setter()"));
      require(success);

      address payable payableA=address(uint160(address(a)));
      return(payableA.send(2 ether));
   }
}
```

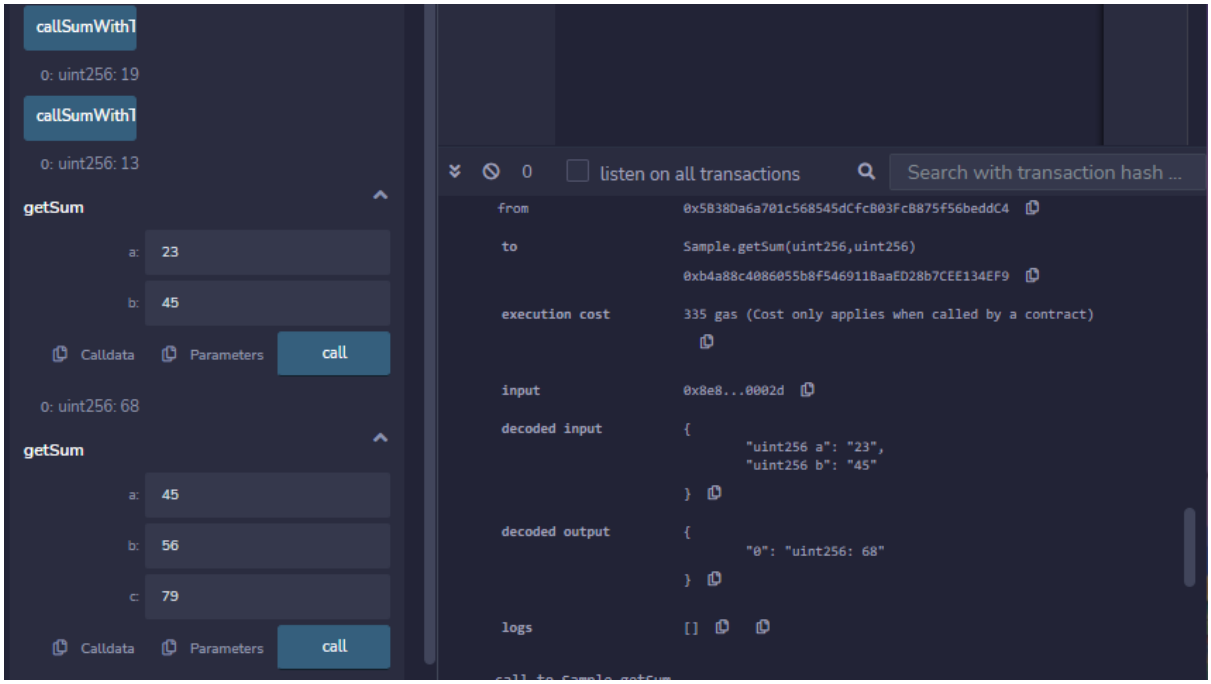
Output:

5. .Function Overloading

Code:

```
pragma solidity ^0.5.12;

contract Sample{
   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 callSumWithTwoArguments() public pure returns (uint){
      return getSum(4,9);
   }

    function callSumWithThreeArguments() public pure returns (uint){
      return getSum(4,9,6);
   }
}
```

Output:

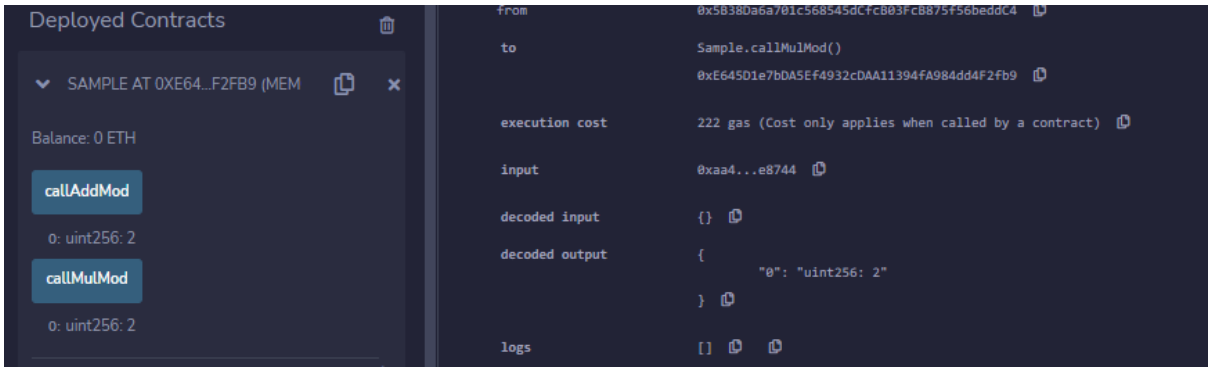


6. .Mathematical Functions

Code:

```
pragma solidity ^0.5.0;

contract Sample{
    function callAddMod() public pure returns (uint){
        return addmod(3,4,5);
//3+4 % 5
    }

   function callMulMod() public pure returns (uint){
        return mulmod(3,4,5);
    }
//3*4 % 5
}
```

Output:



7. .Cryptographic Functions

Code:

```
pragma solidity ^0.5.12;
contract Test{
   function callsha256() public pure returns(bytes32 result){
      return sha256("rifath");
   }
   function callkeccak256() public pure returns(bytes32 result){
      return keccak256("rifath");
   }
}
```

Output:

PRACTICAL 7

Implement and demonstrate the use of the following in Solidity

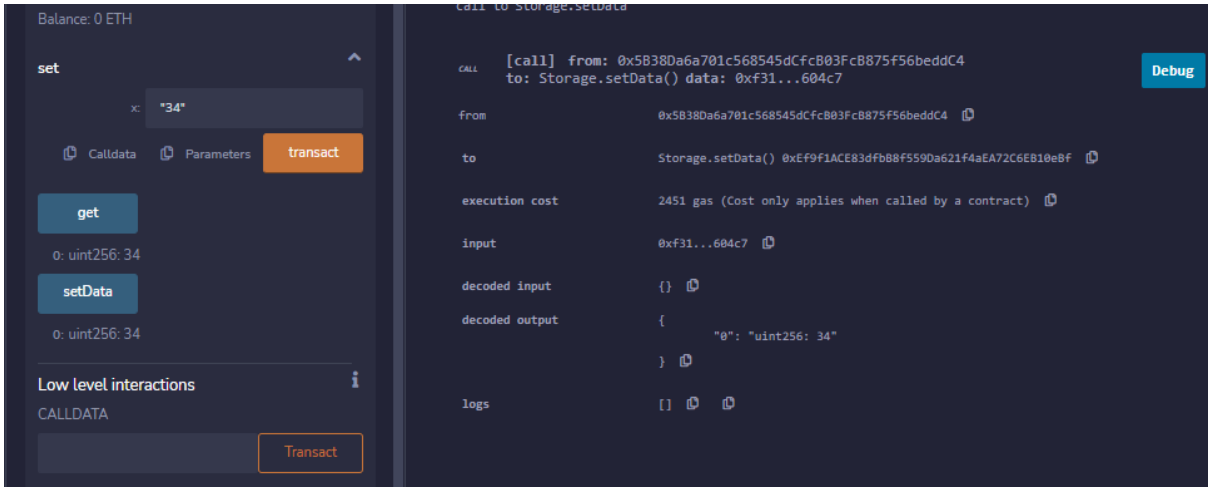
1. Contracts
2. Inheritance
3. Constructors
4. Abstract class
5. Interfaces

1. Contracts

Code:

```
pragma solidity ^0.8.0;
contract Storage
{
uint public setData;
function set(uint x) public{
setData = x;
}
function get() public view returns (uint) {
   return setData;
}}
```

Output:



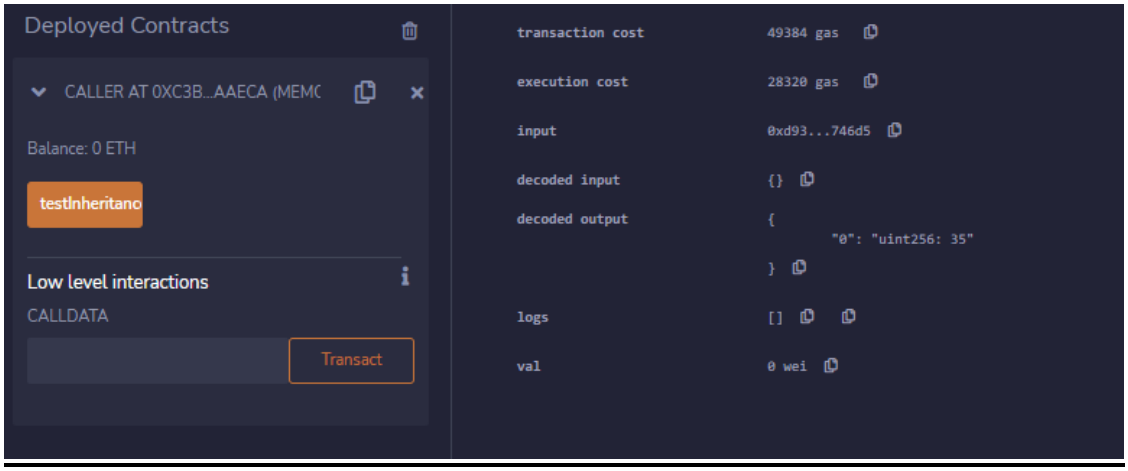
2. Inheritance

a. Single Inheritance:

Code:

```
pragma solidity 0.5.0;
contract parent{

   uint internal sum;

   function setValue() external {
      uint a = 10;
      uint b = 25;
      sum = a + b;
   }
}
contract child is parent{ //defining the child contract

   function getValue(
   ) external view returns(uint) {
      return sum;
   }
}
contract caller {
   child cc = new child();

   function testInheritance(
   ) public returns (uint) {
      cc.setValue();
      return cc.getValue();
   }
}
```

Output:

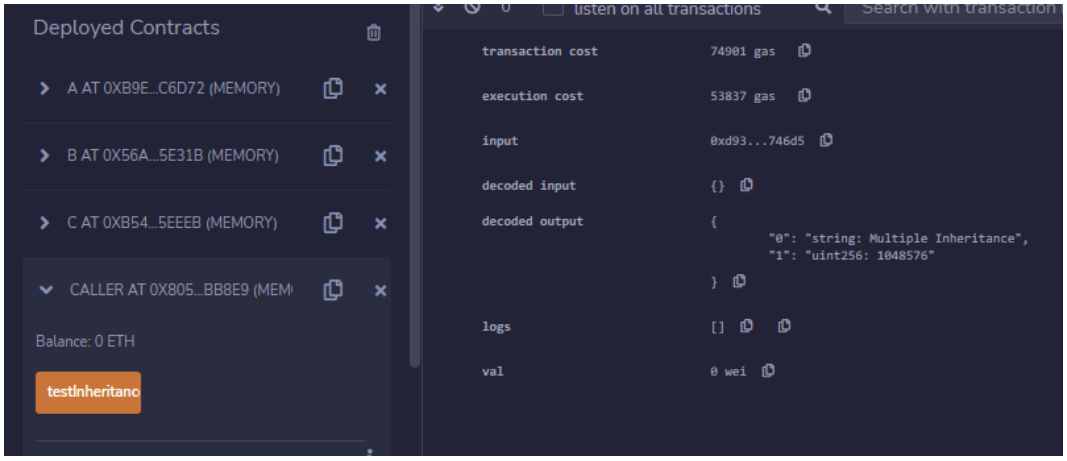


b. **Multiple Inheritance:**

Code:

```
pragma solidity ^0.5.0;

contract A {
    string internal x;

    function setA() external {
        x = "Multiple Inheritance";
    }
}
contract B {
    uint256 internal pow;

    function setB() external {
        uint256 a = 2;
        uint256 b = 20;
        pow = a**b;
    }
}
contract C is A, B {
    function getStr() external view returns (string memory)
    {
        return x;
    }
    function getPow() external view returns (uint256)
    {
        return pow;
    }
}
contract caller {
    C contractC = new C();
    function testInheritance() public returns (string memory, uint256) {
        contractC.setA();
        contractC.setB();
        return (contractC.getStr(), contractC.getPow());
    }
}
```

Output:

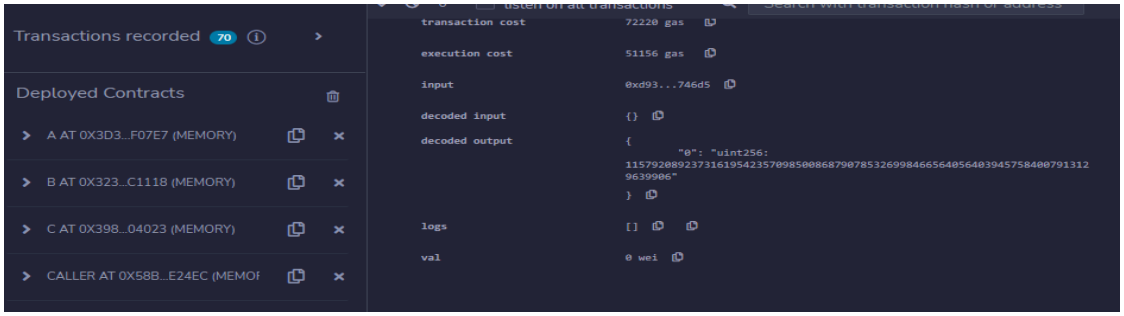


c. **Multilevel Inheritance:**

Code:

```
pragma solidity ^0.5.0;

contract A {
    uint256 internal x;
  function setX() external {
      x=600;
  }
}
contract B is A {
    uint256 internal y;
  function setY() external {
      y=20-x;
  }
}
contract C is B{
  function getY() external view returns(
    uint){
      return y;
  }
}
contract caller {
  C cc = new C();
  function testInheritance(
  ) public returns (
    uint256) {
      cc.setX();
      cc.setY();
      return cc.getY();
  }
}
```

Output:

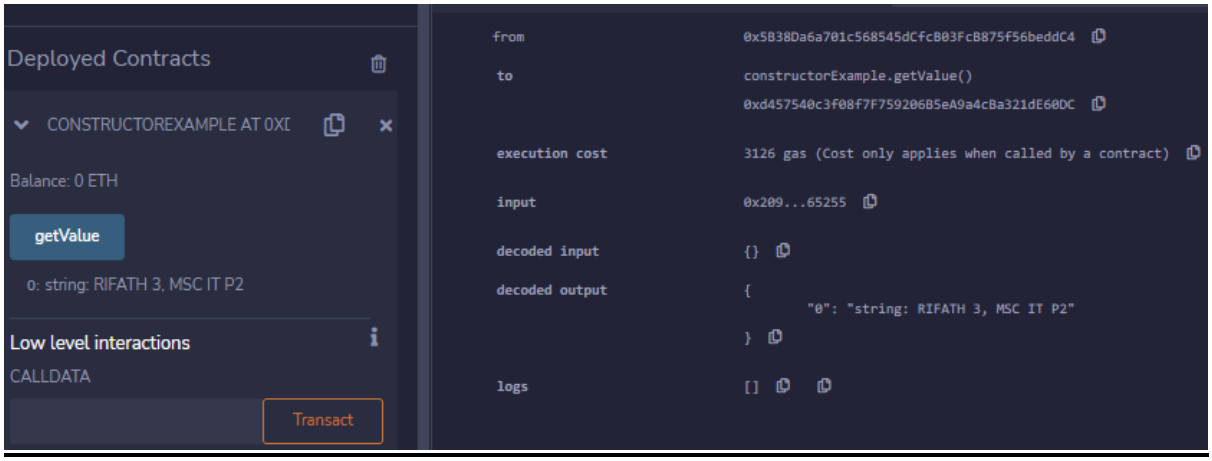


3. Constructors

Code:

```
pragma solidity ^0.5.0;

// Creating a contract
contract constructorExample {

// Declaring state variable
string str;

constructor() public {
str = "RIFATH 3, MSC IT P2";
}
// Defining function to
// return the value of 'str'
function getValue(
) public view returns (
string memory) {
return str;
}
}
```

Output:

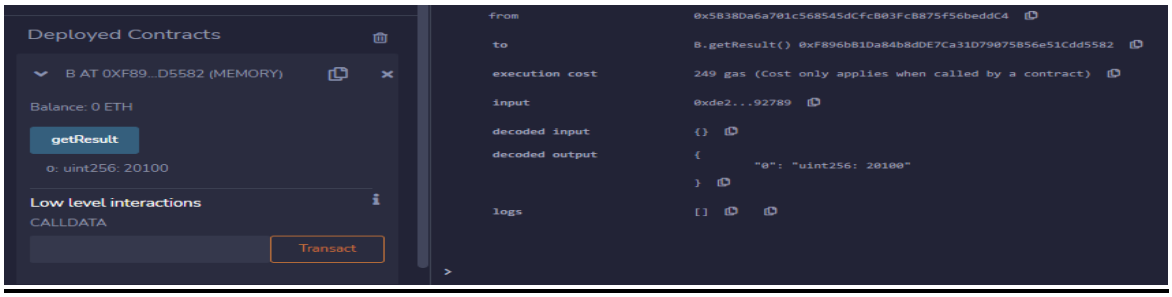


4. Abstract class

Code:

```
pragma solidity ^0.5.0;

contract A {
  function getResult() public view returns(uint);
}
contract B is A {
  function getResult() public view returns(uint) {
    uint a = 100;
    uint b = 201;
    uint result = a * b;
    return result;
  }
}
```

Output:



5. Interfaces

Code:

```
pragma solidity ^0.5.0;

interface Calculator {
  function getResult() external view returns(uint);
}
contract Test is Calculator {
  constructor() public {}
  function getResult() external view returns(uint){
    uint a = 1;
    uint b = 2;
    uint result = a + b;
    return result;
  }
}
```

Output:


Deployed Contracts
TEST AT 0XA3A...5CE5D (MEMOF
Balance: 0 ETH
getResult
0: uint256: 3
Low level interactions
CALLDATA
Transact
from
0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to
Test.getResult() 0xa3A518Ba4e193Fb129aa379F5916d4660f15cE5D
execution cost
247 gas (Cost only applies when called by a contract)
input
0xde2...92789
decoded input
{}
decoded output
{
"0": "uint256: 3"
}
logs
[]

PRACTICAL 8

Implement and demonstrate the use of the following in Solidity

1. Libraries

Code:

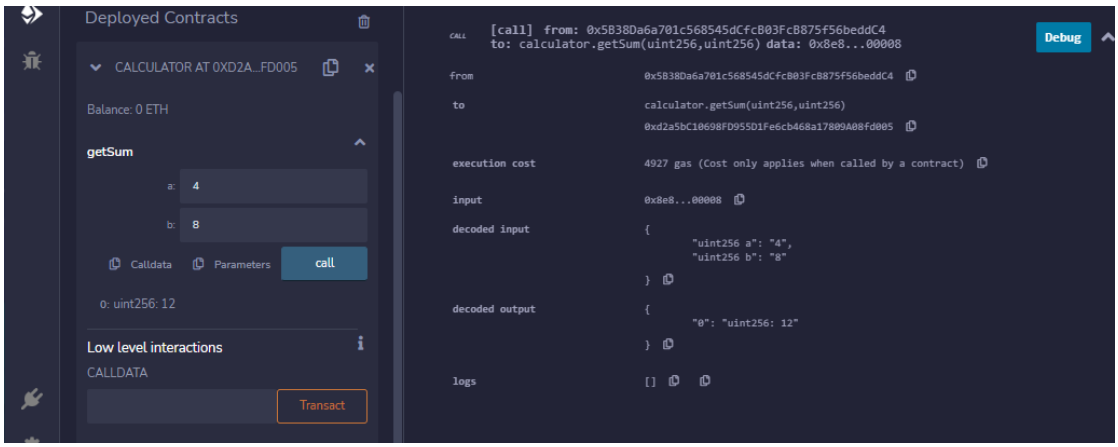
Libraries.sol:

```
pragma solidity ^0.8.0;
import "./MathUtils.sol";
contract calculator{
   using MathUtils for uint;

   function getSum(uint a, uint b) public pure returns(uint){
      return a.add(b);
   }
}
```

MathUtils.sol:

```
pragma solidity ^0.8.0;
library MathUtils{
   function add(uint x, uint y) public pure returns(uint){
      return x+y;
   }
}
```

Output:

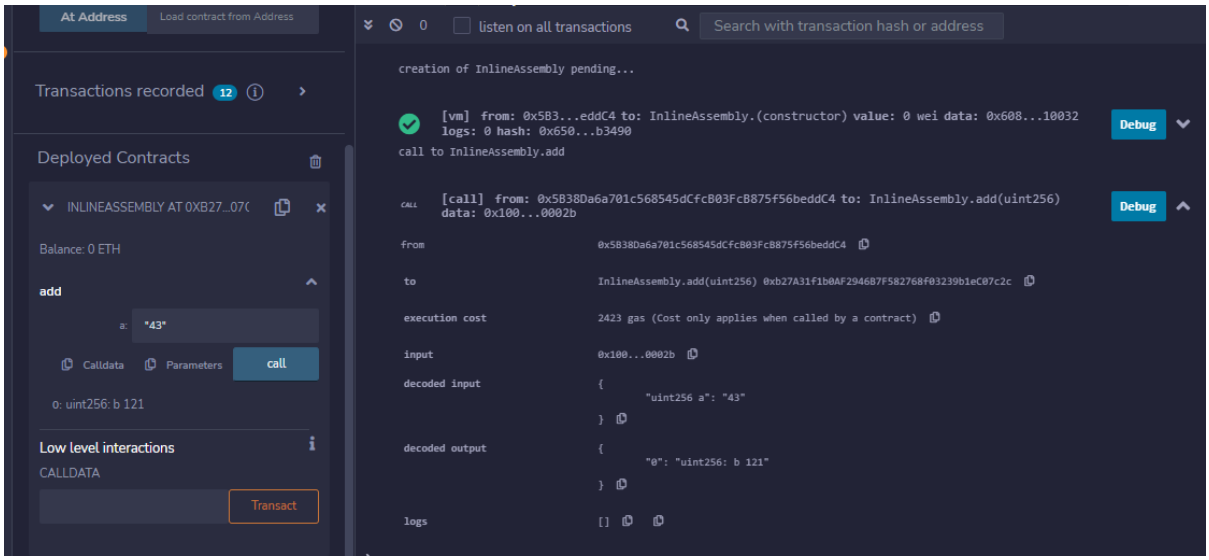


2. Assembly

Code:

```
pragma solidity ^0.5.0;
contract InlineAssembly {
   function add(uint a) public view returns (uint b) {
      assembly {
         let c := add(a, 56)
         mstore(0x80, c)
         {
            let d := add(sload(c), 22)
            b := d
         }
         b := add(b, c)
      }
   }
}
```

Output:

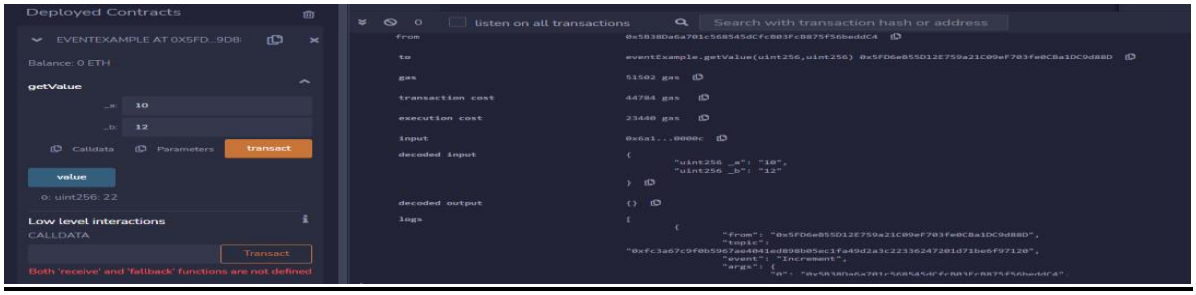


3. Events

Code:

```
pragma solidity ^0.4.21;
contract eventExample {
   uint256 public value = 0;
   event Increment(address owner);
   function getValue(uint _a, uint _b) public { // _a, _b is instance variable (used internally only)
      emit Increment(msg.sender);
      value = _a + _b;
   }
}
```

Output:



4. Error Handling

a. Require
b. Assert
c. Revert

a. Require:

Code:

```
pragma solidity ^0.5.0;
//RIFATH 3
contract requireStatement {

   function checkInput(uint _input) public view returns(string memory){
      require(_input >= 0, "invalid uint8");
      require(_input <= 255, "invalid uint8");
      return "Input is Uint8";
   }
   function Odd(uint _input) public view returns(bool){
      require(_input % 2 != 0);
      return true;
   }
}
```

Output:

b. Assert:

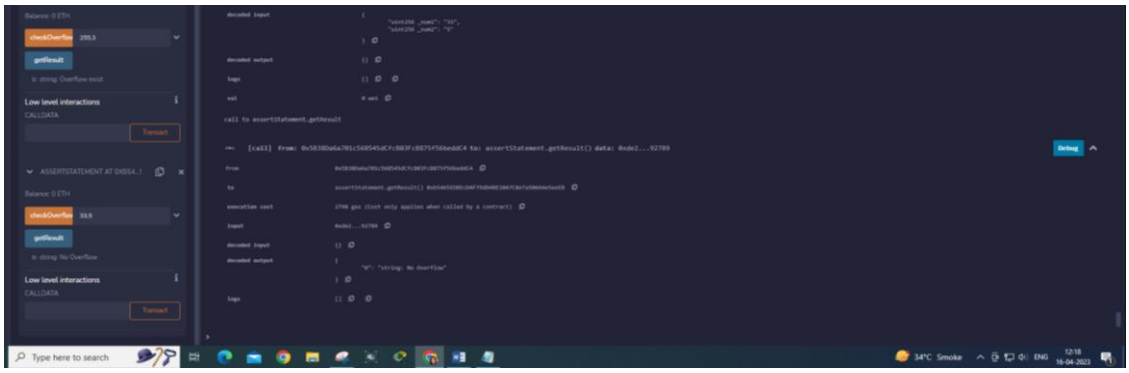
Code:

```
pragma solidity ^0.5.0;
//RIFATH 3
contract assertStatement {
   bool result;
   function checkOverflow(uint _num1, uint _num2) public {
      uint sum = _num1 + _num2;
      assert(sum<=255);
      result = true;
   }

   function getResult() public view returns(string memory){
      if(result == true){
         return "No Overflow";
      }
      else{
         return "Overflow exist";
      }
   }
}
```

Output:

Overflow exists:

No Overflow:

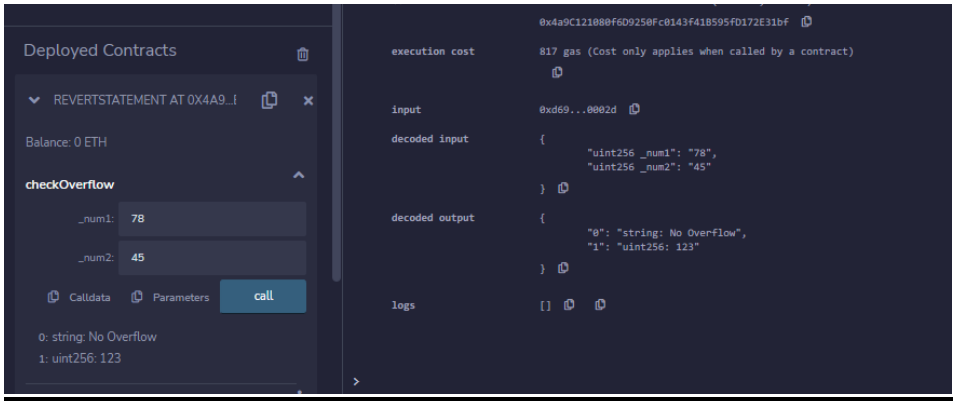


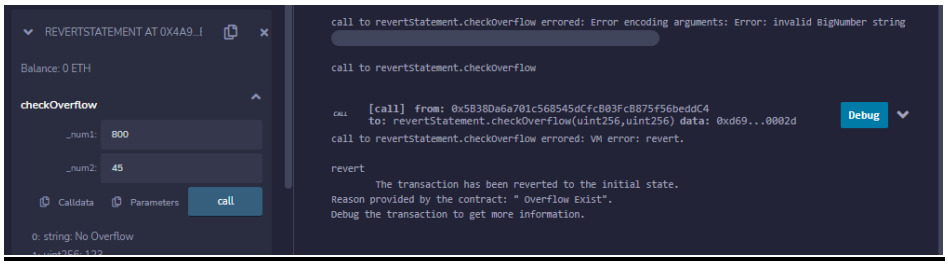
c. **Revert:**

Code:

```
pragma solidity ^0.5.0;
//Rifath 3
contract revertStatement {

function checkOverflow(uint _num1, uint _num2) public view returns(string memory, uint)
{
        uint sum = _num1 + _num2;
        if(sum < 0 || sum > 255){
            revert(" Overflow Exist");
        }
        else{
            return ("No Overflow", sum);
        }
    }
}
```

Output:


Deployed Contracts
REVERTSTATEMENT AT 0X4A9...
Balance: 0 ETH
checkOverflow
_num1: 78
_num2: 45
Calldata
Parameters
call
0: string: No Overflow
1: uint256: 123
0x4a9C121080f6D9250Fc0143f41B595fD172E31bf
execution cost
817 gas (Cost only applies when called by a contract)
input
0xd69...0002d
decoded input
{
"uint256 _num1": "78",
"uint256 _num2": "45"
}
decoded output
{
"0": "string: No Overflow",
"1": "uint256: 123"
}
logs
[]


REVERTSTATEMENT AT 0X4A9...
Balance: 0 ETH
checkOverflow
_num1: 800
_num2: 45
Calldata
Parameters
call
0: string: No Overflow
call to revertStatement.checkOverflow errored: Error encoding arguments: Error: invalid BigNumber string
call to revertStatement.checkOverflow
CALL
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to: revertStatement.checkOverflow(uint256,uint256) data: 0xd69...0002d
Debug
call to revertStatement.checkOverflow errored: VM error: revert.
revert
The transaction has been reverted to the initial state.
Reason provided by the contract: " Overflow Exist".
Debug the transaction to get more information.