PRACTICAL NO 3:

Implement and demonstrate the use of the following in Solidity :

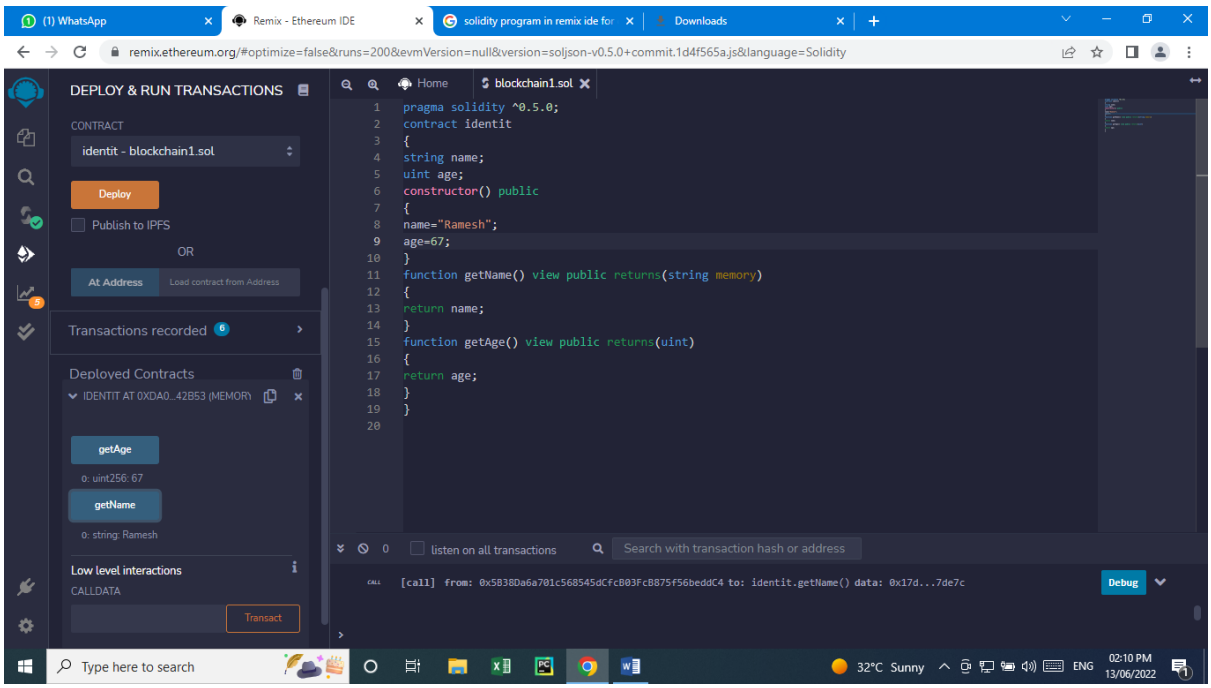
A] Variables, Operators, Loops, Decision Making, Strings, Arrays, Enums, Structs, Mappings.

1. Variables :

CODE :

```
pragma solidity ^0.5.0;
contract identity
  {
    string name;
    uint age;
    constructor() public
    {
      name="Salman";
      age=57;
    }
    function getName() view public returns(string memory)
    {
      return name;
    }
    function getAge() view public returns(uint)
    {
      return age;
    }
  }
```

OUTPUT :

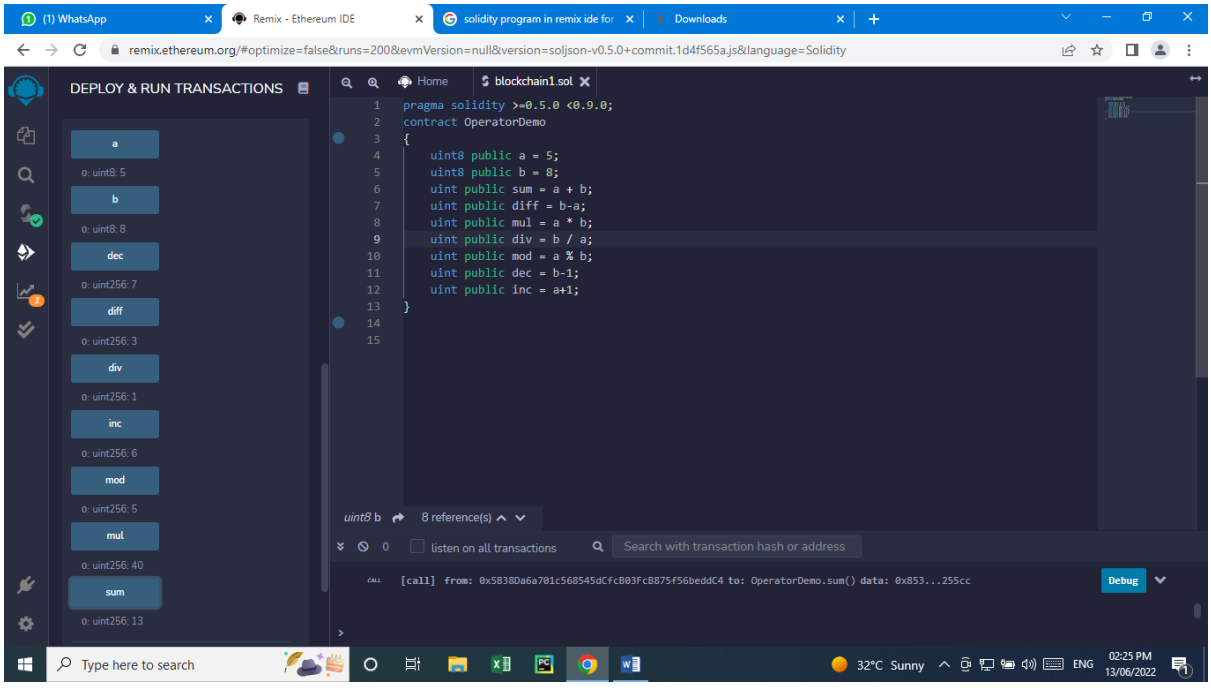


2. Operators :

CODE :

```
pragma solidity >=0.5.0 <0.9.0;
contract OperatorDemo
{
   // Initializing variables
   uint16 public a = 20;
   uint16 public b = 10;

   // Initializing a variable
   // with sum
   uint public sum = a + b;

   // with the difference
   uint public diff = a - b;

   // with product
   uint public mul = a * b;

   // with quotient
   uint public div = a / b;

   // with modulus
   uint public mod = a % b;

   // decrement value
   uint public dec = --b;

   // with increment value
   uint public inc = ++a;
}
```

OUTPUT :



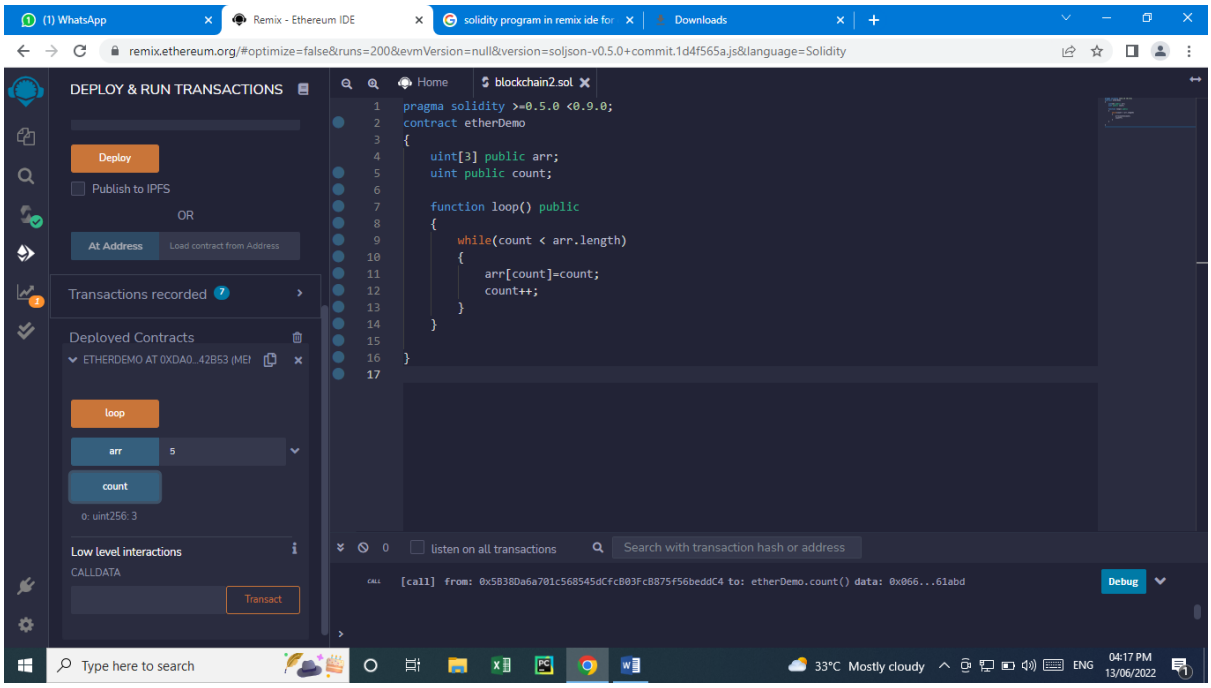
3. Loops :

While loop :-

CODE :

```
pragma solidity >=0.5.0 <0.9.0;

contract etherDemo
{
   uint[3] public arr;
   uint public count;

   function loop() public
   {
      while(count < arr.length)
      {
         arr[count]=count;
         count++;
      }
   }

}
```

OUTPUT :



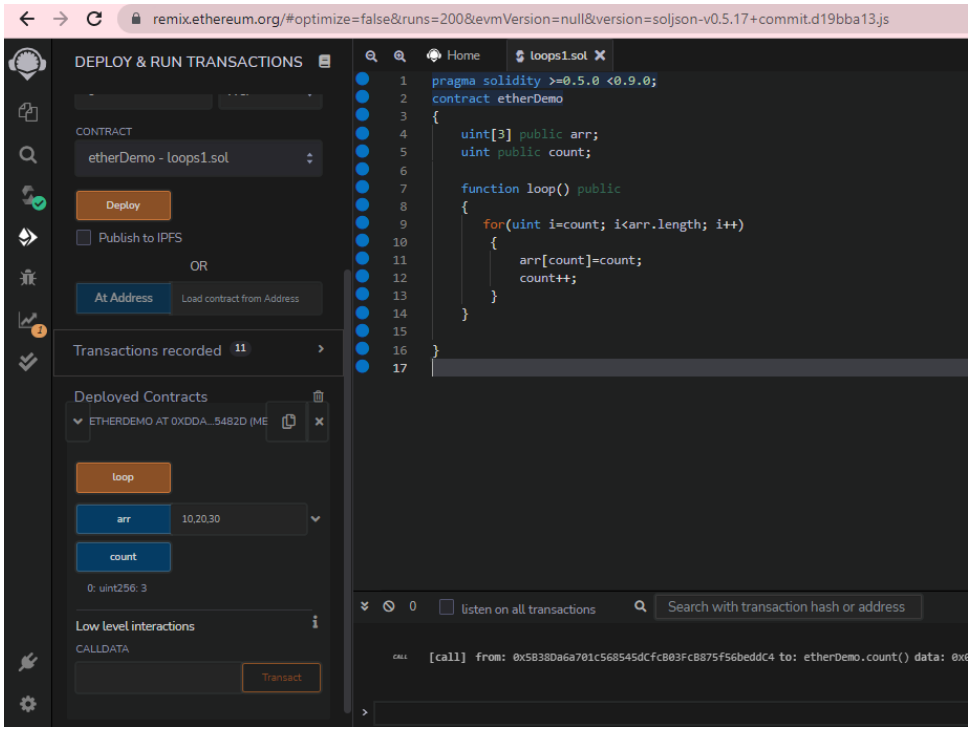
For loop :-

CODE :

```
pragma solidity >=0.5.0 <0.9.0;
contract etherDemo
{
   uint[3] public arr;
   uint public count;
```

```
  function loop() public
  {
    for(uint i=count; i<arr.length; i++)
    {
       arr[count]=count;
       count++;
    }
  }

}
```

OUTPUT :



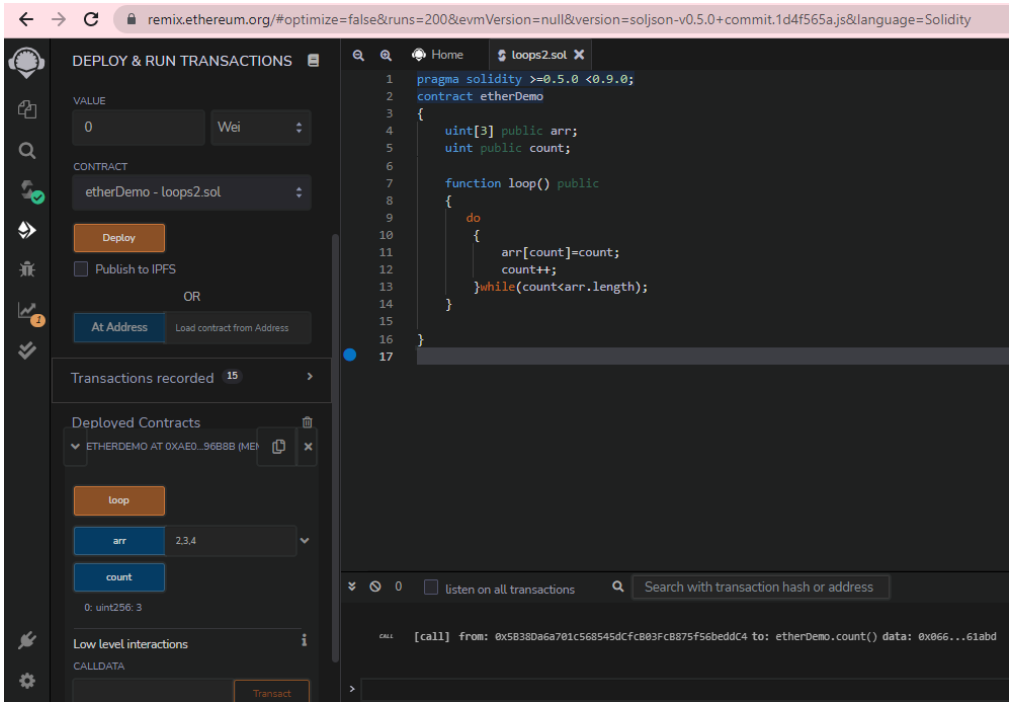
Do-while loop :-

CODE :

```
pragma solidity >=0.5.0 <0.9.0;
contract etherDemo
{
  uint[3] public arr;
  uint public count;

  function loop() public
  {
    do
    {
       arr[count]=count;
       count++;
    }while(count<arr.length);
  }
```

```
}
```

OUTPUT :



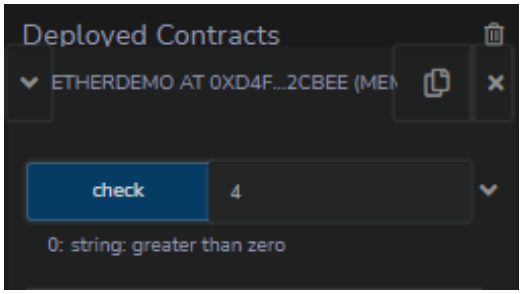
Decision Making :

CODE :

```
pragma solidity >=0.5.0 <0.9.0;
contract etherDemo
{
  function check(int a) public pure returns(string memory)
  {
     string memory value;
     if(a>0)
     {
        value="greater than zero";
     }
     else if(a==0)
     {
        value="equal to zero";
     }
     return value;
  }
}
```

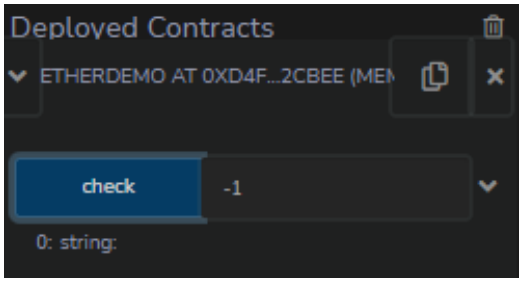
OUTPUT :

Greater :



Equal :

Less :



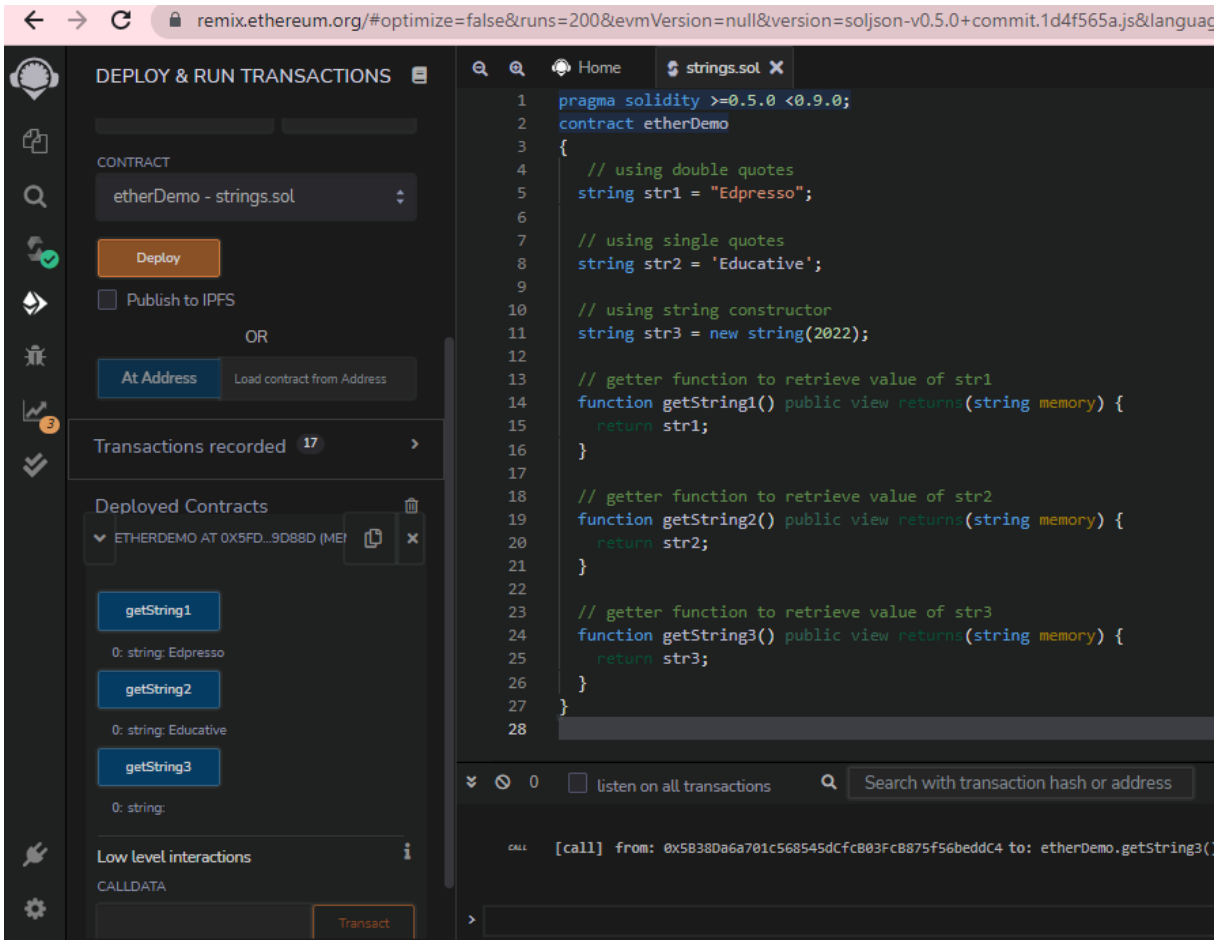
Strings :

CODE :

```
pragma solidity >=0.5.0 <0.9.0;
contract etherDemo
{
  // using double quotes
 string str1 = "Edpresso";

 // using single quotes
 string str2 = 'Educative';

 // using string constructor
 string str3 = new string(2022);

 // getter function to retrieve value of str1
 function getString1() public view returns(string memory) {
   return str1;
 }

 // getter function to retrieve value of str2
 function getString2() public view returns(string memory) {
   return str2;
 }
```

```
 // getter function to retrieve value of str3
 function getString3() public view returns(string memory) {
   return str3;
 }
}
```

OUTPUT :



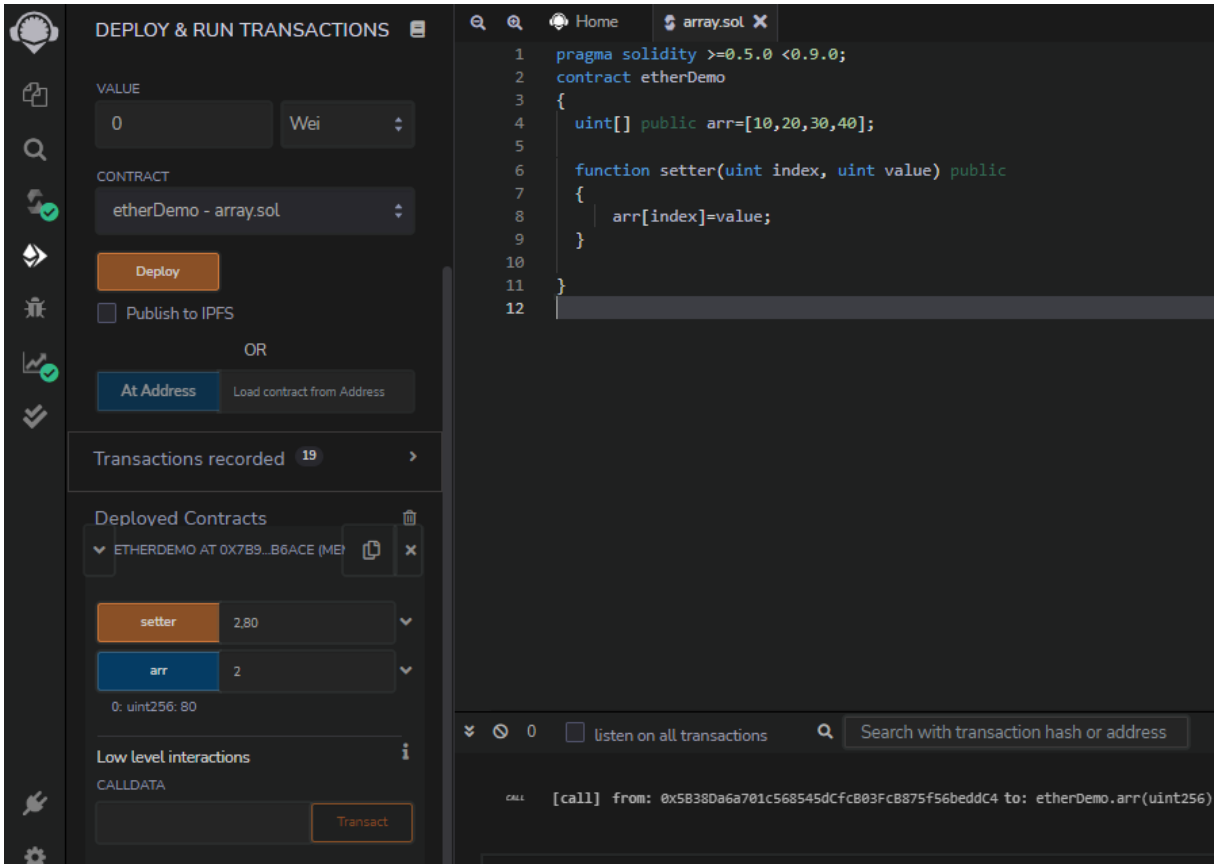
Arrays :

Fixed length:-

CODE :

```
pragma solidity >=0.5.0 <0.9.0;
contract etherDemo
{
 uint[] public arr=[10,20,30,40];

 function setter(uint index, uint value) public
 {
    arr[index]=value;
 }

}
```

OUTPUT :



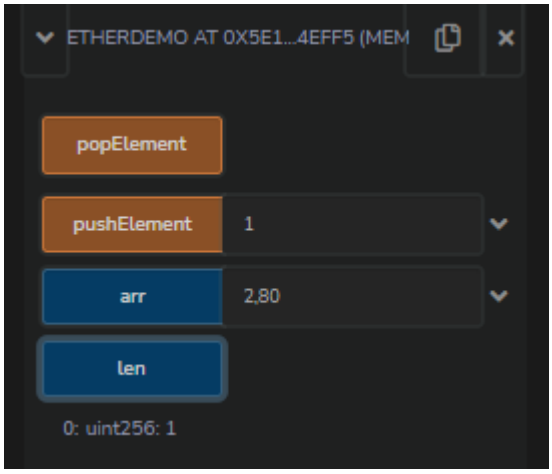
Dynamic length :-

CODE :

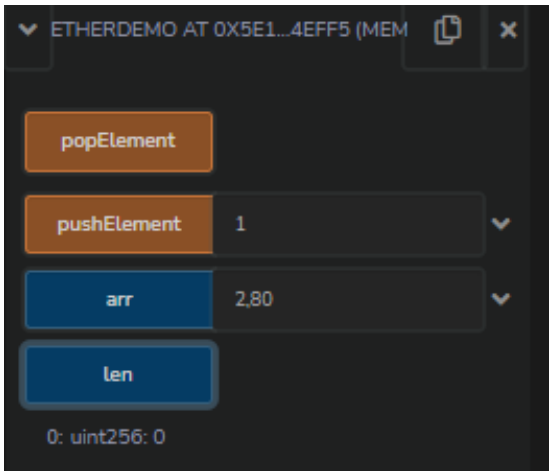
```
pragma solidity >=0.5.0 <0.9.0;
contract etherDemo
{
 uint[] public arr;
 function pushElement(uint item) public
 {
  arr.push(item);
 }
 function len()public view returns(uint)
 {
  return arr.length;
 }
 function popElement()public
 {
  arr.pop();
 }
}
```

OUTPUT :

Push element :-



Pop element :-

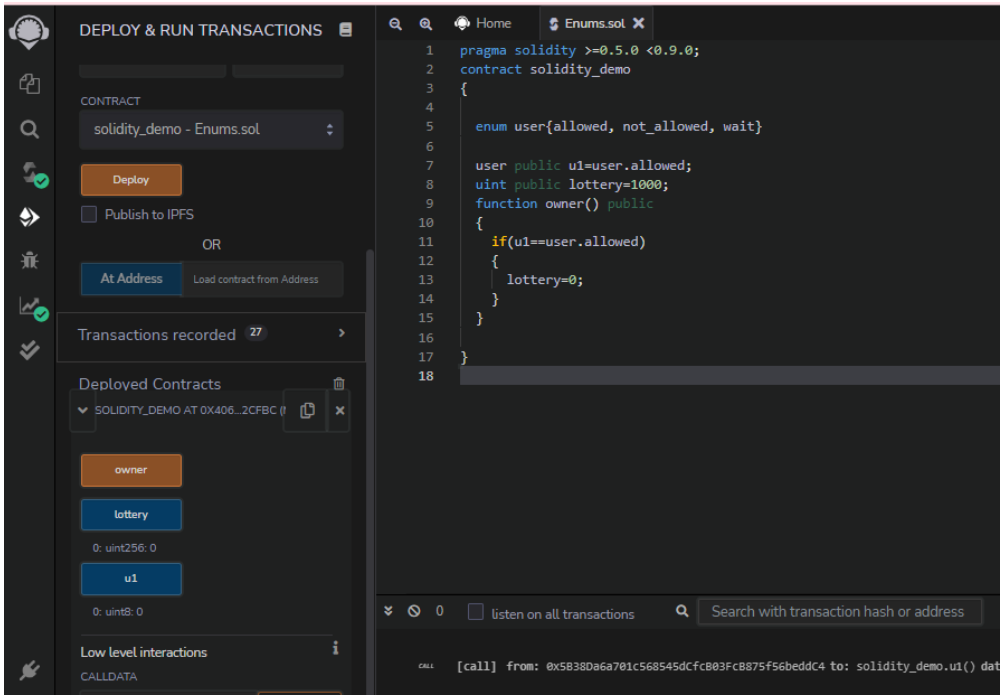


Enums :

CODE :

```
pragma solidity >=0.5.0 <0.9.0;
contract solidity_demo
{

 enum user{allowed, not_allowed, wait}

 user public u1=user.allowed;
 uint public lottery=1000;
 function owner() public
 {
  if(u1==user.allowed)
  {
    lottery=0;
  }
 }

}
```

OUTPUT :



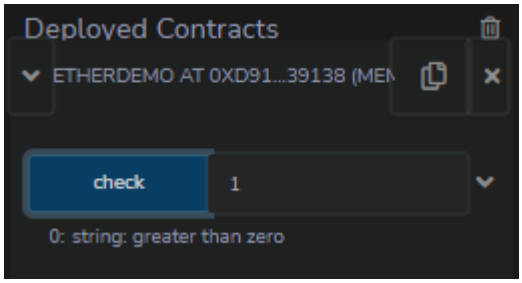
Structs :

CODE :

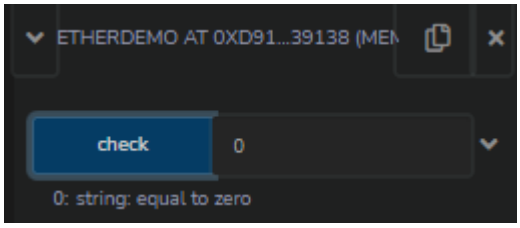
```
pragma solidity >=0.5.0 <0.9.0;

 struct Student
  {
    uint roll;
    string name;
  }

contract solidity_demo
{
  Student public s1;

  constructor(uint _roll, string memory _name)
  {
    s1.roll=_roll;
    s1.name=_name;
  }

}
```

OUTPUT :

A greater than zero :



A equal to zero :

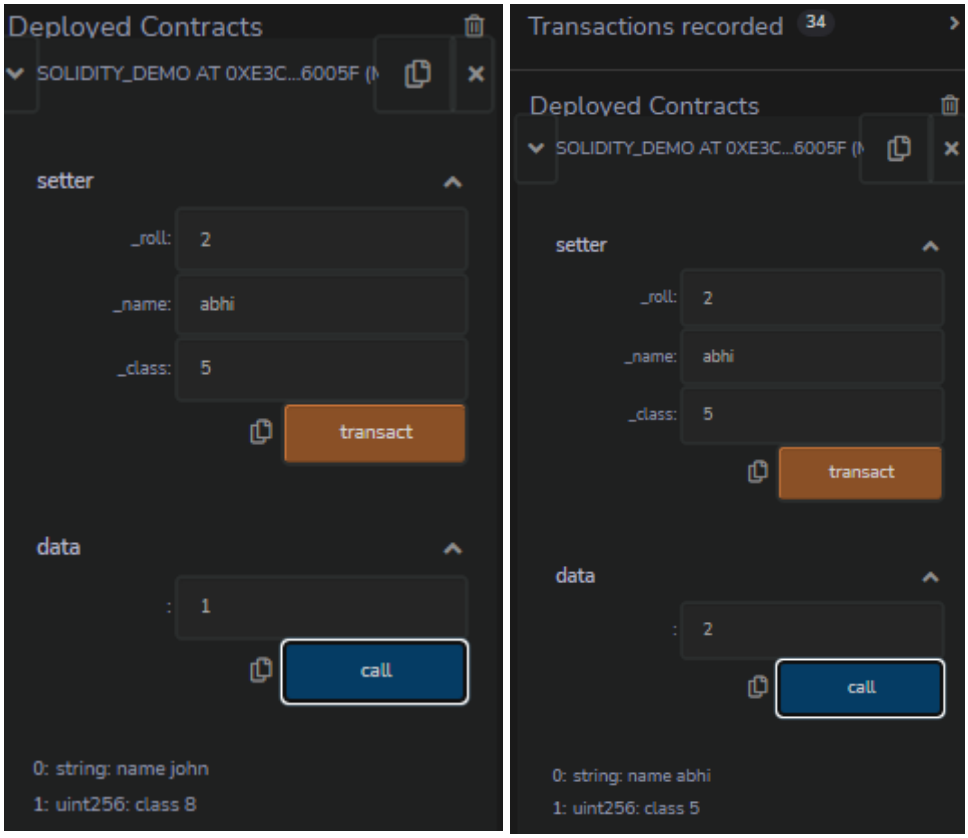


Mappings :

CODE :

```
pragma solidity >=0.5.0 <0.9.0;
contract solidity_demo
{
 struct Student
 {
   string name;
   uint class;
 }
 mapping (uint => Student) public data;

 function setter(uint _roll, string memory _name, uint _class) public
 {
   data[_roll]= Student(_name, _class);
 }

}
```

OUTPUT :



B] Functions, Function Modifiers, View functions, Pure functions, Fallback

Functions **:**

CODE :

```
pragma solidity >=0.5.0 <0.9.0;
contract solidity_demo
{
  uint age=10;
  function getter() public view returns(uint)
  {
    return age;
  }
  function setter() public
  {
    age=age+1;
  }

}
```

OUTPUT :

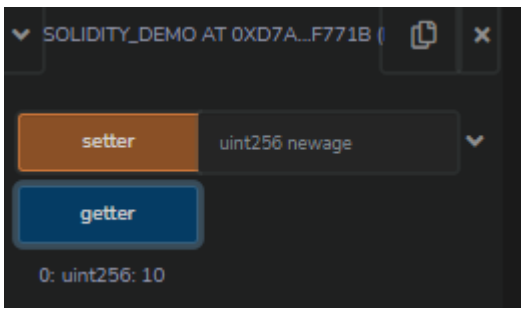
View functions:

CODE :

```
pragma solidity >=0.5.0 <0.9.0;
contract solidity_demo
{
  uint age=10;
  function getter() public view returns(uint)
  {
    return age;
  }
  function setter(uint newage) public
  {
    age=newage;
  }
}
```

OUTPUT :

Defined age = 10:



Age after set :

Pure Functions:

CODE :

```
pragma solidity >=0.5.0 <0.9.0;
contract solidity_demo
{
  uint public age=10;
  function getter() public pure returns(uint)
  {
    uint roll=100;
    return roll;
  }

}
```

OUTPUT :

Function overloading :

CODE :

```
pragma solidity >=0.5.0 <0.9.0;
contract solidity_demo
{
   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;
    }

}
```

OUTPUT :

Mathematical functions :

CODE :

```
pragma solidity >=0.5.0 <0.9.0;
contract solidity_demo
{
    function callAddMod() public pure returns(uint){
        return addmod(4, 5, 3);
    }
    function callMulMod() public pure returns(uint){
        return mulmod(4, 5, 3);
    }

}
```

OUTPUT :

Cryptographic functions :

CODE :

```
pragma solidity >=0.5.0 <0.9.0;
contract solidity_demo
{
    function callKeccak256() public pure returns(bytes32 result)
    {
      return keccak256("ABC");
    }
    function callripemd160() public pure returns(bytes20 result)
    {
      return ripemd160("ABC");
    }
    function callsha256() public pure returns(bytes32 result)
    {
      return sha256("ABC");
    }
}
```

OUTPUT :