PRACTICAL 7

[CS601] – Cryptography and Blockchain

Date – 06/02/2023 | *By* Aishwarya Suryakant Waghmare, PRN – 2001106059

Title/Aim of the practical :

To write a program to create following contracts in solidity:

Array

// Exercise create a function that can fully remove an item from an array

// Create an Empty array called changeArray

// Create a function called removeElement which sets the index argument of the array to the last element in the array

// Remove the last index from that function with the pop method

// Create a function called test which pushes 1 2 3 4 into changeArray

// Remove the element 2 from the array when the contract is called

Mapping

// Exercise 1 – Deploy the mapping contract, create some keys as addresses, and set those keys to unique values

// 2. Remove all the addresses and check to see their updated value.

// Mapping Assignment:

// Create a unique data type as a struct called Movie and give it the string properties: title and diretor.

// Create a map called movie which takes a uint as a key and Movie as a value

// Create a function called addMovie which takes three inputs, movie id, title and director which assigns a value of an integer to a movie added back to the movie map. It should include a title and director name.

// Deploy the contract and update the movie information to the movie map with our favourite movies!

Apparatus/Tools/ Resources used :

- Lecture Notes
- E-Resources
- E-Book
- Laptop

Procedure of the practical :

To write a program to create following contracts in solidity:

Array

// Exercise create a function that can fully remove an item from an array

// Create an Empty array called changeArray

// Create a function called removeElement which sets the index argument of the array to the last element in the array

// Remove the last index from that function with the pop method

// Create a function called test which pushes 1 2 3 4 into changeArray

// Remove the element 2 from the array when the contract is called

Mapping

// Exercise 1 – Deploy the mapping contract, create some keys as addresses, and set those keys to unique values

// 2. Remove all the addresses and check to see their updated value.

// Mapping Assignment:

// Create a unique data type as a struct called Movie and give it the string properties: title and diretor.

// Create a map called movie which takes a uint as a key and Movie as a value

// Create a function called addMovie which takes three inputs, movie id, title and director which assigns a value of an integer to a movie added back to the movie map. It should include a title and director name.

// Deploy the contract and update the movie information to the movie map with our favourite movies!

Program Code :

Arrays

```
pragma solidity 0.5.0;
contract learnarrays{


  uint[] public myArray;
  uint[20] public myfixedArray;
  function push(uint number) public{
    myArray.push(number);
  }
  function pop() public{
    myArray.pop();
  }
  function getlength() public view returns(uint){
    return myArray.length;
  }
}


// Solidity program to demonstrate
// Pop operation
pragma solidity ^0.5.0;
```

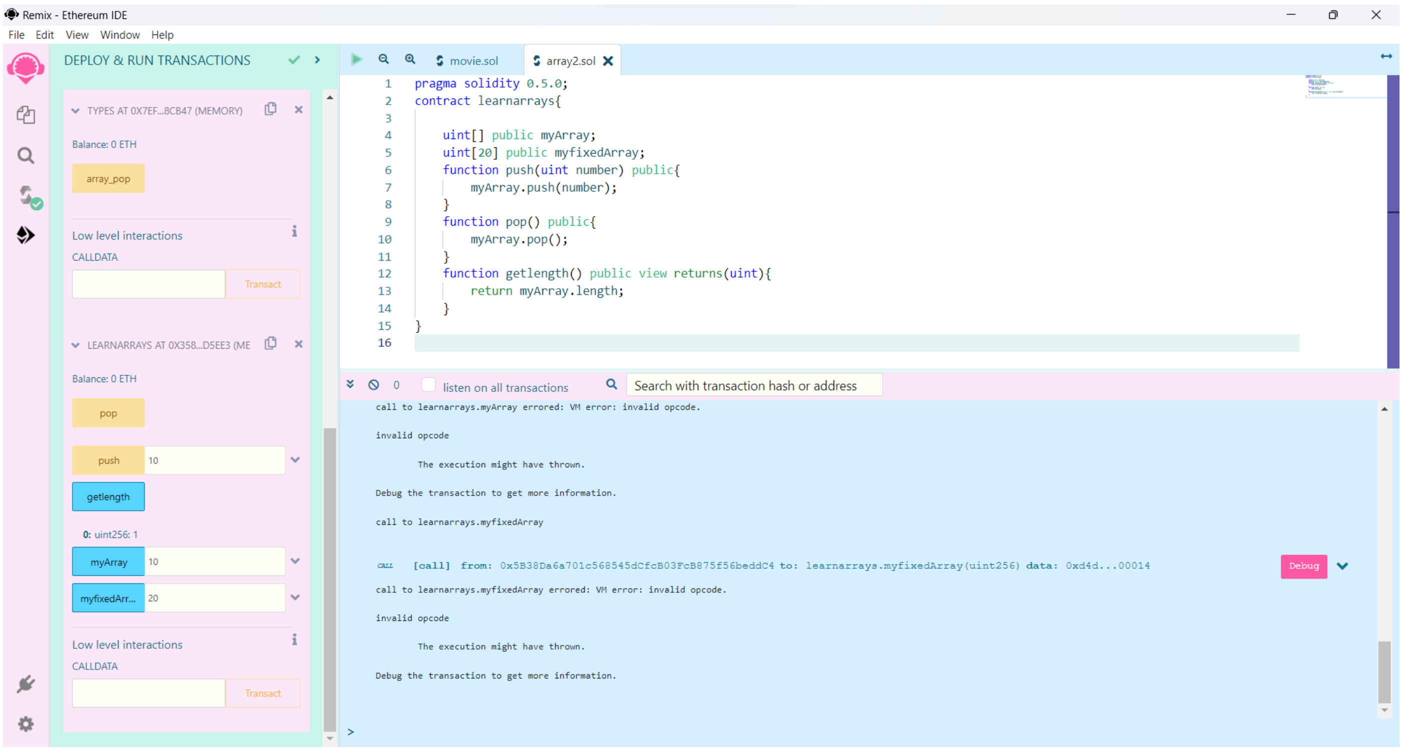
```
// Creating a contract
contract Types {

   // Defining an array
   uint[] data
     = [10, 20, 30, 40, 50];

   // Defining a function to
   // pop values from the array
   function array_pop(
   ) public returns(uint[] memory){
      data.pop();
      return data;
   }
}
```

Mapping

```
pragma solidity ^0.5.0;

contract learnmapping {
  mapping(address=>uint) public myMap;

  function getAddress(address _addr) public view returns(uint){
    return myMap[ _addr];
  }
  function set(address _addr,uint _i) public{
     myMap[_addr]=_i;
  }
}
```

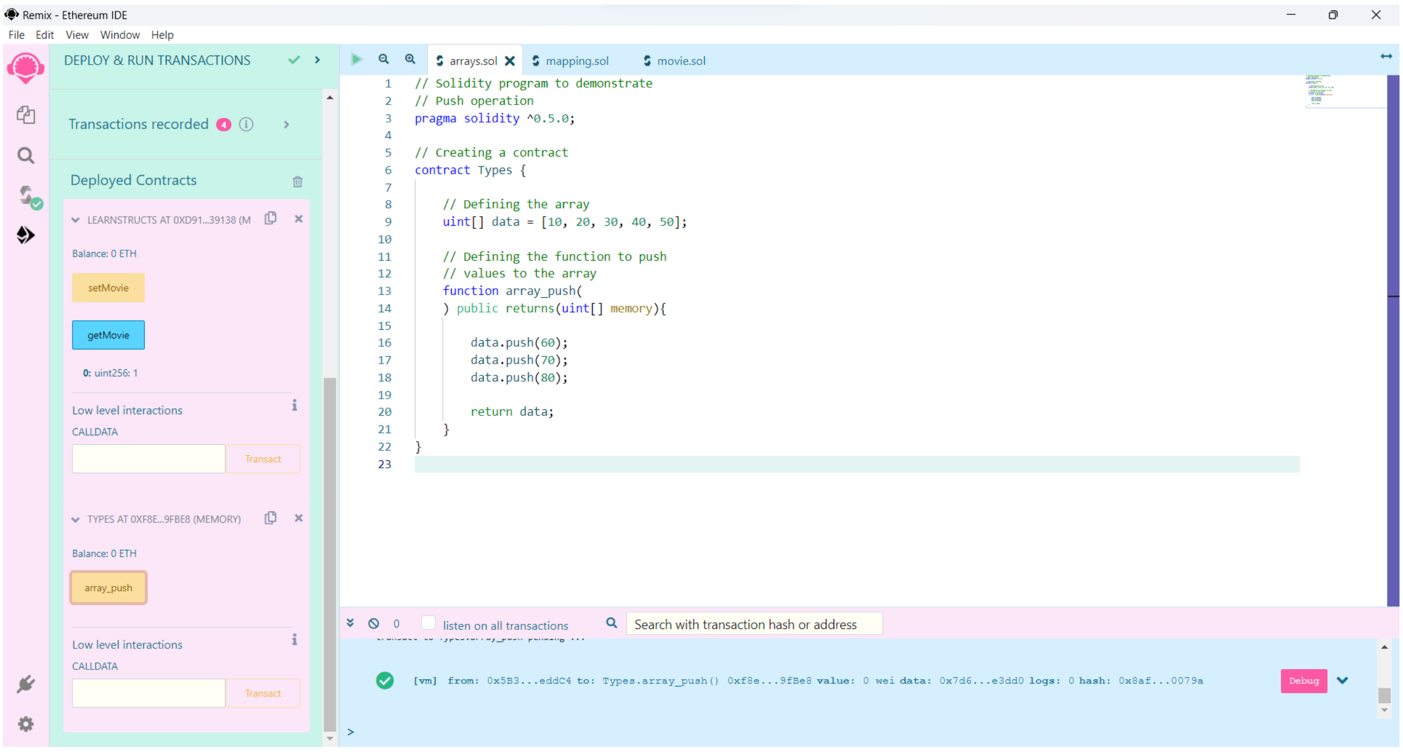
Result/ Output/ Screenshots of the Practical :

Arrays



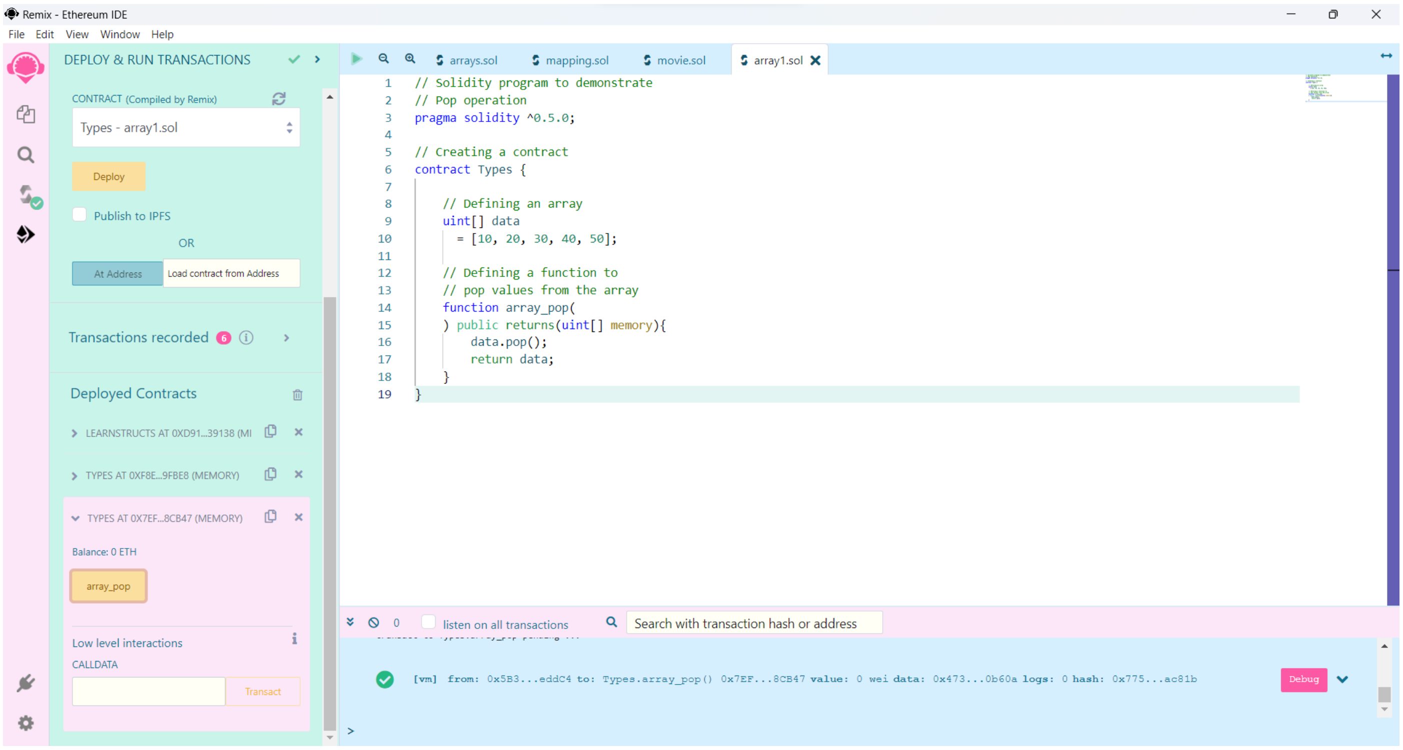
```
pragma solidity 0.5.0;
contract learnarrays{

    uint[] public myArray;
    uint[20] public myfixedArray;
    function push(uint number) public{
        myArray.push(number);
    }
    function pop() public{
        myArray.pop();
    }
    function getlength() public view returns(uint){
        return myArray.length;
    }
}
```

```
call to learnarrays.myArray errored: VM error: invalid opcode.

invalid opcode

        The execution might have thrown.

Debug the transaction to get more information.

call to learnarrays.myfixedArray

CALL  [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: learnarrays.myfixedArray(uint256) data: 0xd4d...00014
call to learnarrays.myfixedArray errored: VM error: invalid opcode.

invalid opcode

        The execution might have thrown.

Debug the transaction to get more information.

>
```



```
// Solidity program to demonstrate
// Push operation
pragma solidity ^0.5.0;

// Creating a contract
contract Types {

    // Defining the array
    uint[] data = [10, 20, 30, 40, 50];

    // Defining the function to push
    // values to the array
    function array_push(
    ) public returns(uint[] memory){

        data.push(60);
        data.push(70);
        data.push(80);

        return data;
    }
}
```

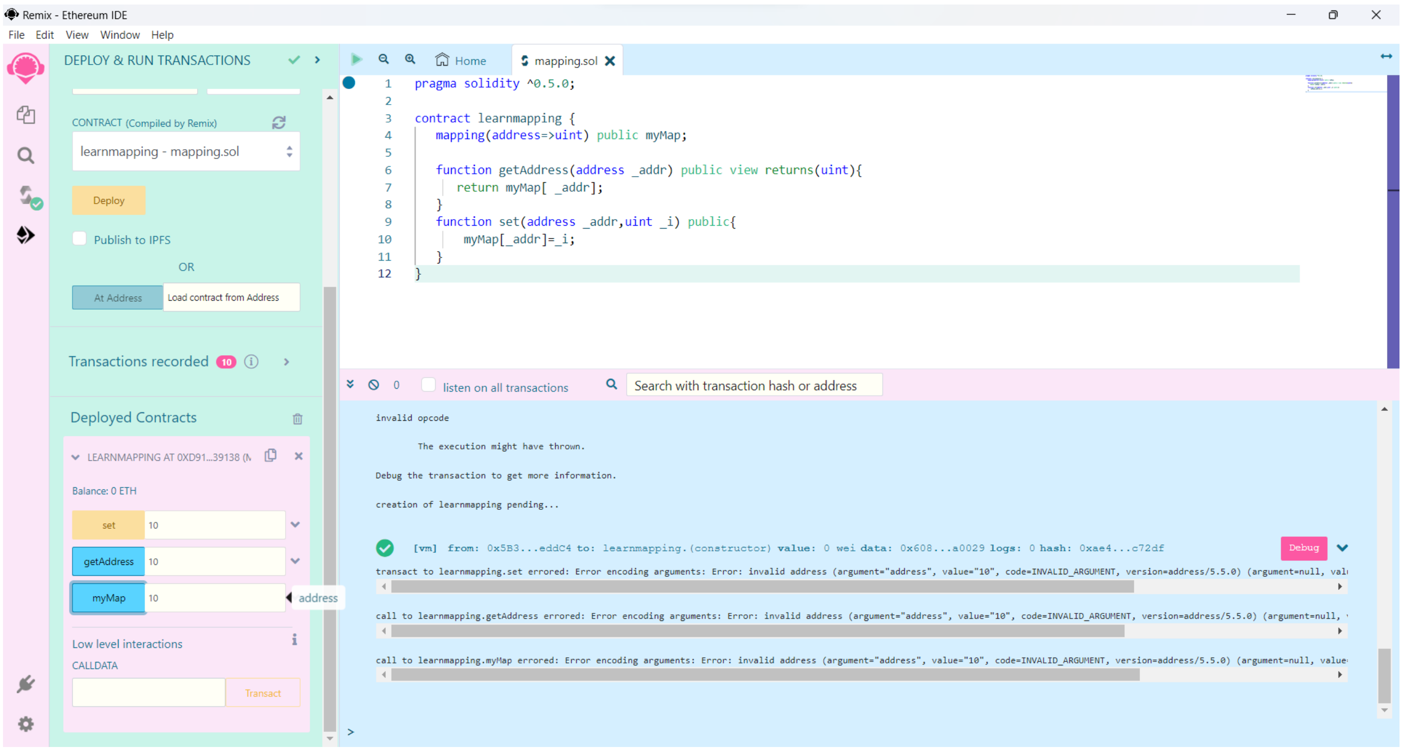
```
[vm] from: 0x5B3...eddC4 to: Types.array_push() 0xf8e...9fBe8 value: 0 wei data: 0x7d6...e3dd0 logs: 0 hash: 0x8af...0079a

>
```



```
// Solidity program to demonstrate
// Pop operation
pragma solidity ^0.5.0;

// Creating a contract
contract Types {

    // Defining an array
    uint[] data
      = [10, 20, 30, 40, 50];

    // Defining a function to
    // pop values from the array
    function array_pop(
    ) public returns(uint[] memory){
        data.pop();
        return data;
    }
}
```

```
[vm] from: 0x5B3...eddC4 to: Types.array_pop() 0x7EF...8CB47 value: 0 wei data: 0x473...0b60a logs: 0 hash: 0x775...ac81b
```

Mapping



```
pragma solidity ^0.5.0;

contract learnmapping {
   mapping(address=>uint) public myMap;

   function getAddress(address _addr) public view returns(uint){
      return myMap[ _addr];
   }
   function set(address _addr,uint _i) public{
       myMap[_addr]=_i;
   }
}
```

```
invalid opcode

        The execution might have thrown.

Debug the transaction to get more information.

creation of learnmapping pending...

[vm] from: 0x5B3...eddC4 to: learnmapping.(constructor) value: 0 wei data: 0x608...a0029 logs: 0 hash: 0xae4...c72df
transact to learnmapping.set errored: Error encoding arguments: Error: invalid address (argument="address", value="10", code=INVALID_ARGUMENT, version=address/5.5.0) (argument=null, val

call to learnmapping.getAddress errored: Error encoding arguments: Error: invalid address (argument="address", value="10", code=INVALID_ARGUMENT, version=address/5.5.0) (argument=null,

call to learnmapping.myMap errored: Error encoding arguments: Error: invalid address (argument="address", value="10", code=INVALID_ARGUMENT, version=address/5.5.0) (argument=null, value
```

Parameters achieved/ Conclusion :

Therefore, understood and wrote program in the solidity as well as created the smart contracts for the arrays and mapping.