

PRACTICAL 6

[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:

Looping Practice

```
// Create a contract myLoopingPracticeContract and place all the following code within:  
// Create a list that ranges from 1 to 20 called longList  
// Create a list called numbersList of the following numbers: 1, 4, 34, 56  
// Create a function that loops through numbersList and returns a true value if the number that the user  
inputs exist in the list otherwise it should return false.  
// Create a function that loops through and returns how many even numbers there are in the long list
```

Enum

```
// Create an enum for the color of shirts called shirtColor and set it to the options of either RED or WHITE  
or BLUE  
// Create a data of shirtColor called defaultChoice which is a constant set to the color BLUE  
// Create a data of shirtColor called choice and don't initiate the value  
// Create a function called setWhite which changes the shirt color of shirtColor to white  
// Create a function getChoice which returns the current choice of shirtColor  
// Create a function getDefaultChoice which returns the default choice of shirtColor
```

Structs

```
// Create a new movie and set it up so that it updates to the movie in the setMovie function  
// Return the id of the new movie  
// Create a new var called comedy and set up comedy to the datatype Movie  
// Update the setMovie function with a comedy movie that contain name, director, and an id  
// Return the movie id of the comedy.
```

Exercises with Strings

```
// Create a string called favoriteColor  
// Set the favorite color of the string favoriteColor to blue  
// Create a function which returns the string literal of favoriteColor  
// Create a function which changes the favoriteColor string literal from blue to your favorite color.  
// Create a function which can return how many characters there are in the string favorite color
```

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:

Looping Practice

// Create a contract myLoopingPracticeContract and place all the following code within:

// Create a list that ranges from 1 to 20 called longList

// Create a list called numbersList of the following numbers: 1, 4, 34, 56

// Create a function that loops through numbersList and returns a true value if the number that the user inputs exist in the list otherwise it should return false.

// Create a function that loops through and returns how many even numbers there are in the long list

Enum

// Create an enum for the color of shirts called shirtColor and set it to the options of either RED or WHITE or BLUE

// Create a data of shirtColor called defaultChoice which is a constant set to the color BLUE

// Create a data of shirtColor called choice and do not initiate the value

// Create a function called setWhite which changes the shirt color of shirtColor to white

// Create a function getChoice which returns the current choice of shirtColor

// Create a function getDefaultChoice which returns the default choice of shirtColor

Structs

// Create a new movie and set it up so that it updates to the movie in the setMovie function

// Return the id of the new movie

// Create a new var called comedy and set up comedy to the datatype Movie

// Update the setMovie function with a comedy movie that contain name, director, and an id

// Return the movie id of the comedy.

Exercises with Strings

// Create a string called favoriteColor

// Set the favorite color of the string favoriteColor to blue

// Create a function which returns the string literal of favoriteColor

// Create a function which changes the favoriteColor string literal from blue to your favourite color.

// Create a function which can return how many characters there are in the string favourite color

Looping Practice

```
pragma solidity ^0.5.0;
```

```
contract myLoopingContract {
```

```
    uint [] longlist = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20];
```

```
    uint [] numbersList = [1,4,34,56];
```

```
    function checkMultiples1(uint number) public view returns (uint){
```

```
        uint count=0;
```

```
        for(uint i=1; i< numbersList.length; i++){
```

```
            if(checkMultipleValidity1(numbersList[i], number)) {
```

```
                count++;
```

```
            }
```

```
        }
```

```
        return count;
```

```
    }
```

```
    function checkMultipleValidity1(uint num, uint nums) public view returns (bool){
```

```
        if(num % nums == 0){
```

```
            return true;
```

```
        } else{
```

```
            return false;
```

```
        }
```

```
    }
```

```
    function checkEven(uint number) public view returns (uint){
```

```
        uint count=0;
```

```
        for(uint i=1; i< longlist.length; i++){
```

```
            if(checkMultipleValidity1(longlist[i], 2)) {
```

```
                count++;
```

```
            }}
```

```
        return count;
```

```
    }
```

```
}
```

Enum

```
pragma solidity ^0.5.0;

contract learnenum {
    enum shirtcolor{ RED, WHITE, BLUE }
    shirtcolor choice;
    shirtcolor constant defaultChoice = shirtcolor.BLUE;

    function setWHITE() public {
        choice = shirtcolor.WHITE;
    }
    function getChoice() public view returns (shirtcolor) {
        return choice;
    }
    function getDefaultChoice() public pure returns (uint) {
        return uint(defaultChoice);
    }
}
```

Structs

```
pragma solidity ^0.5.0;
contract learnstructs {
    struct Movie {
        string title;
        string director;
        uint movie_id;
    }
    Movie Comedy;

    function setMovie() public {
        Comedy = Movie('Cirkus', 'Robert', 1);
    }
    function getMovie() public view returns(uint){
        return Comedy.movie_id;
    }
}
```

String Exercises

```
pragma solidity ^0.5.0;

contract learnstrings {

    // Hello is a string literal
    string greetings = 'hello!';

    function sayHello() public view returns (string memory){
        return greetings;
    }

    function changeGreetings(string memory change) public {
        greetings = change;
    }

    function getStringLength() public view returns(uint){
        bytes memory stringToBytes = bytes(greetings);
        return stringToBytes.length;
    }

    string favoriteColor = 'blue';

    function getColor() public view returns (string memory) {
        return favoriteColor;
    }

    function changeColor(string memory _color) public {
        favoriteColor = _color;
    }

    function getColorLength() public view returns(uint) {
        bytes memory colorToBytes = bytes(favoriteColor);
        return colorToBytes.length;
    }
}
```

// A list of number ranging from 1 to 10

```
uint [] public numbersList = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
```

```
function checkMultiples(uint _number) public view returns (uint) {
```

```
    uint count = 0;
```

```
    for(uint i = 1; i < numbersList.length; i++){
```

```
        if(checkMultipleValidity(numbersList[i], _number)) {
```

```
            count++;
```

```
        }
```

```
    }
```

```
    return count;
```

```
}
```

```
function checkMultipleValidity(uint _num, uint _nums) public view returns (bool) {
```

```
    if(_num % _nums == 0){
```

```
        return true;
```

```
    } else {
```

```
        return false;
```

```
    }
```

```
}}
```

Result/ Output/ Screenshots of the Practical :

Looping Practice

The screenshot displays the Remix Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel shows the contract 'myLoopingContract' compiled by Remix. The 'Deploy' button is highlighted. Below it, the 'Transactions recorded' section shows a list of transactions. The 'Deployed Contracts' section shows the contract deployed at address 0xD91...3. On the right, the 'looping_practice.sol' file is open, showing the Solidity code. The code defines a contract 'myLoopingContract' with two functions: 'checkMultiples1' and 'checkMultipleValidity1'. The 'checkMultiples1' function iterates through a list of numbers and counts how many are multiples of a given number. The 'checkMultipleValidity1' function checks if a number is divisible by another. The bottom panel shows the transaction logs, indicating that the contract was deployed successfully and the 'checkMultiples1' function was called with the value 256, returning 5. The 'checkMultipleValidity1' function is also called, showing an error in the original image.

Enum

The screenshot displays the Remix IDE interface with a contract named `enum.sol` open. The contract defines an enum `shirtcolor` with values `RED`, `WHITE`, and `BLUE`. It includes functions `setWHITE()`, `getChoice()`, and `getDefaultChoice()`. The left sidebar shows the 'DEPLOY & RUN TRANSACTIONS' panel with a deployed contract `TEST AT 0XD8B...33FA8 (MEMORY)`. The bottom panel shows a list of transactions, including the contract's creation and subsequent calls to `setLarge()`, `getChoice()`, and `getDefaultChoice()`.

```
1 pragma solidity ^0.5.0;
2
3 contract learnenum {
4     enum shirtcolor { RED, WHITE, BLUE }
5     shirtcolor choice;
6     shirtcolor constant defaultChoice = shirtcolor.BLUE;
7
8     function setWHITE() public {
9         choice = shirtcolor.WHITE;
10    }
11    function getChoice() public view returns (shirtcolor) {
12        return choice;
13    }
14    function getDefaultChoice() public pure returns (uint) {
15        return uint(defaultChoice);
16    }
17 }
```

Structs

The screenshot displays the Remix IDE interface with a contract named `movie.sol` open. The contract defines a struct `Movie` with fields `title`, `director`, and `movie_id`. It includes functions `setMovie()` and `getMovie()`. The left sidebar shows the 'DEPLOY & RUN TRANSACTIONS' panel with a deployed contract `LEARNSTRUCTS AT 0XD7A...F771B (M)`. The bottom panel shows a list of transactions, including the contract's creation and subsequent calls to `setMovie()` and `getMovie()`.

```
1 pragma solidity ^0.5.0;
2 contract learnstructs {
3     struct Movie {
4         string title;
5         string director;
6         uint movie_id;
7     }
8     Movie Comedy;
9
10    function setMovie() public {
11        Comedy = Movie('Cirkus', 'Robert', 1);
12    }
13
14    function getMovie() public view returns(uint){
15        return Comedy.movie_id;
16    }
17 }
```

String Exercises

The screenshot displays the Remix Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel shows the deployment of a smart contract named 'LEARNSTRINGS AT 0xDA0...42B53 (M)'. The contract's state is visible, including a balance of 0 ETH and various variables like 'color' (orange), 'greetings' (blue), and 'numbersList' (10). The 'Low level interactions' section shows the 'CALLDATA' field and a 'Transact' button.

The main editor displays the Solidity code for the 'learnstrings' contract:

```
1 pragma solidity ^0.5.0;
2
3 contract learnstrings {
4
5     // Hello is a string literal
6     string greetings = 'hello!';
7
8     function sayHello() public view returns (string memory){
9         return greetings;
10    }
11
12    function changeGreetings(string memory change) public {
13        greetings = change;
14    }
15
16    function getStringLength() public view returns(uint){
17        bytes memory stringToBytes = bytes(greetings);
18        return stringToBytes.length;
19    }
20 }
```

The bottom panel shows the transaction history and debug information. It lists two transactions: a call to 'learnstrings.numbersList' and a call to 'learnstrings.sayHello()'. The first transaction resulted in an 'invalid opcode' error, while the second transaction was successful.

Parameters achieved/ Conclusion :

Therefore, understood and wrote program in the solidity as well as created the smart contracts for the looping practice, enum, strings and structs.