Introduction To Blockchain
Assignment - 1

Patil Amit Gurusidhappa
19104004
B11

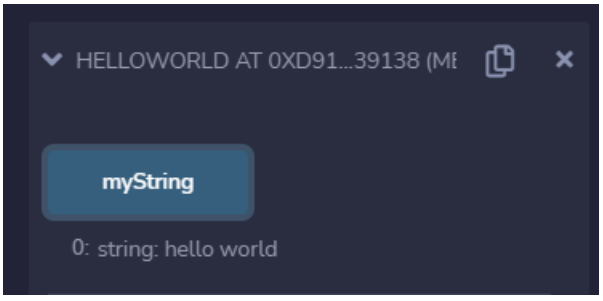
Q1. What is Solidity and Write is basic syntax.

1. Solidity is an object-oriented, high-level language for implementing smart contracts. Smart contracts are programs which govern the behavior of accounts within the Ethereum state.
2. Solidity is a curly-bracket language designed to target the Ethereum Virtual Machine (EVM)
3. Solidity is statically typed, supports inheritance, libraries and complex user-defined types among other features.
4. With Solidity you can create contracts for uses such as voting, crowdfunding, blind auctions, and multi-signature wallets.

Basic Syntax

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.7;
contract HelloWorld{
    string public  myString = "hello world";
}
```

Output



Q2. Write the syntax of the following

i) Array Syntax

```
type arrayName [ arraySize ];
```

ii) Enum Syntax

```
enum <enumerator_name> {
            element 1, element 2,....,element n
}
```

iii) Struct Syntax

```
struct <structure_name> {
   <data type> variable_1;
   <data type> variable_2;
}
```

Q3. Explain the data types available in Solidity.

Type	Keyword	Values
Boolean	bool	true/false
Integer	int/uint	Signed and unsigned integers of varying sizes.
Integer	int8 to int256	Signed int from 8 bits to 256 bits. int256 is the same as int.
Integer	uint8 to uint256	Unsigned int from 8 bits to 256 bits. uint256 is the same as uint.
Fixed Point Numbers	fixed/unfixed	Signed and unsigned fixed point numbers of varying sizes.
Fixed Point Numbers	fixed/unfixed	Signed and unsigned fixed point numbers of varying sizes.

Q4. Explain the concept of Mapping in Solidity.

Mapping in Solidity acts like a hash table or dictionary in any other language. These are used to store the data in the form of key-value pairs, a key can be any of the built-in data types but reference types are not allowed while the value can be of any type. Mappings are mostly used to associate the unique Ethereum address with the associated value type.

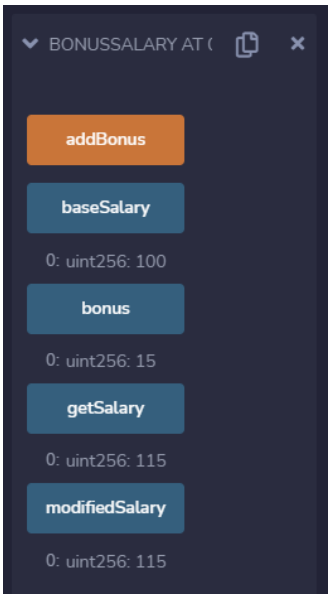
Syntax:

```
mapping(key => value) <access specifier> <name>;
```

Q5. Ashish is an accountant in head and shoulder company, As it was Christmas eve all the employees got bouns salary. Help Ashish to write a smart contract in order to calculate the bonus.

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.7;

contract BonusSalary {
    uint public baseSalary=100;
    uint public bonus=15;
    uint public modifiedSalary=100;

    event ModifiedSalary(uint value);

    function getSalary()view public returns(uint){

        return modifiedSalary;
    }

    function addBonus()public{
        modifiedSalary=baseSalary+bonus;
        // modifiedSalary+=1;
        emit ModifiedSalary(modifiedSalary);
    }
}
```

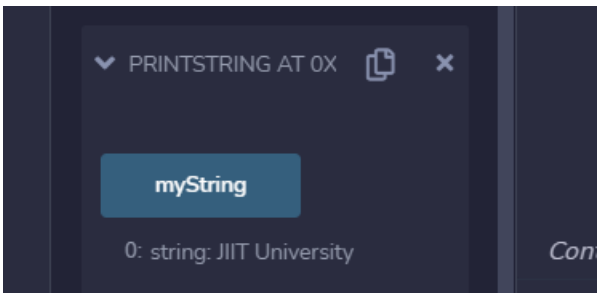
Output



Q6. Junaid started learning solidity language, He completed his theory classes now it is time for practical session. Junaid was given a problem to print the string "JIIT University". Help Junaid to print the String.

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.7;
contract PrintString{
    string public  myString = "JIIT University";
}
```

Output

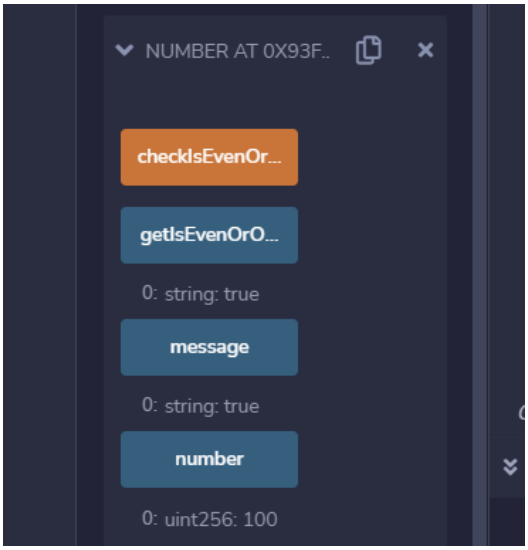


Q7. Sinchan is a student in Kasukabey city school in Japan, His teacher gave him homework to check whether a number is Even or Odd number, Help Sinchan to solve the problem.

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.7;

contract Number {
    uint public number=100;
```

```
    string public message;

    function getIsEvenOrOdd()view public returns(string memory){

        return message;
    }

    function checkIsEvenOrOdd()public{
       if(number%2==0){
           message="true";
       }else{
           message="false";
       }
    }

}
```

Output



Q8. Write the Syntax of the following with example.
i) Inheritance ii) Enums iii) Struct
i) Array

Syntax

```
type arrayName [ arraySize ];
```

Example

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.7;
```

```
contract arrayTest {
   function testArray() public pure{
      uint len = 7;

      //dynamic array
      uint[] memory a = new uint[](7);

      //bytes is same as byte[]
      bytes memory b = new bytes(len);

      assert(a.length == 7);
      assert(b.length == len);

      //access array variable
      a[6] = 8;

      //test array variable
      assert(a[6] == 8);

      //static array
      uint[3] memory c = [uint(1) , 2, 3];
      assert(c.length == 3);
   }
}
```

ii) Enum

Syntax

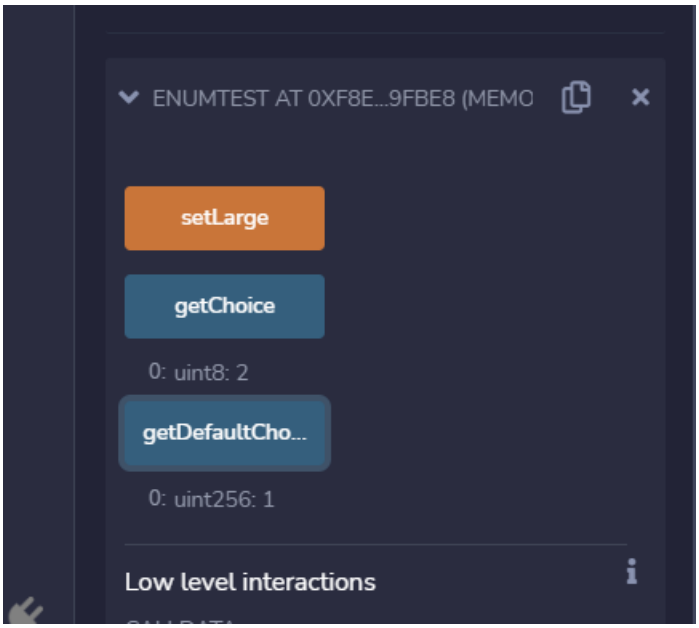
```
enum <enumerator_name> {
            element 1, element 2,....,element n
}
```

Example

```
contract enumtest {
   enum FreshJuiceSize{ SMALL, MEDIUM, LARGE }
   FreshJuiceSize choice;
   FreshJuiceSize constant defaultChoice = FreshJuiceSize.MEDIUM;

   function setLarge() public {
      choice = FreshJuiceSize.LARGE;
   }
   function getChoice() public view returns (FreshJuiceSize) {
      return choice;
```

```
    }
    function getDefaultChoice() public pure returns (uint) {
        return uint(defaultChoice);
    }
}
```

Output



iii) Struct

Syntax

```
struct <structure_name> {
    <data type> variable_1;
    <data type> variable_2;
}
```

Example

```
contract structTest {
    struct Book {
        string title;
        string author;
        uint book_id;
    }
    Book book;

    function setBook() public {
```

```
        book = Book('Learn Java', 'TP', 1);
    }
    function getBookId() public view returns (uint) {
        return book.book_id;
    }
}
```

Output