Vaneet Singla
102015136
3NC6

Name: Vaneet Singla Class:3NC6 Roll no.:102015136

ASSIGNMENT-1

1. **Create a file in solidity to declare variables of different data types and arrays (fixed dynamic) and use a function to get their values.**

```
pragma solidity >=0.7.0 <0.9.0;

contract DataTypes {
    // Different Datatypes
    string public str;
    uint256 public num;
    bool public Bool;
    // Array
    uint256[] public arr1;
    //uint256[] public arr2=[1,5,9];

    // Constructor
    constructor() public {
        num = 100;
        str = "Vaneet Singla";
        Bool = true;

        //inserting arr values
        arr1.push(1);
        arr1.push(2);
        arr1.push(3);
    }

    function getString() public view returns(string memory) {
        return str;
    }
    function getNum() public view returns(uint256) {
        return num;
    }
    function getArr() public view returns(uint256[] memory) {
        return arr1;
    }
    function getBool() public view returns(bool) {
        return Bool;
    }
}
```

Vaneet Singla
102015136
3NC6

2. Create a file in solidity to declare functions and experiment with its scope as (public/private, pure/view and returns/no-returns.

```
pragma solidity >=0.7.0 <0.9.0;

contract MyContract {
    // Public function
    function publicFunc() public returns(uint256) {
        return 10;
    }
    // Private function
    function privateFunc() private returns(bool) {
        return true;
    }
    // Pure function
    function pureFunc(uint256 a, uint256 b) pure returns(uint256) {
        return a + b;
```

Vaneet Singla
102015136
3NC6

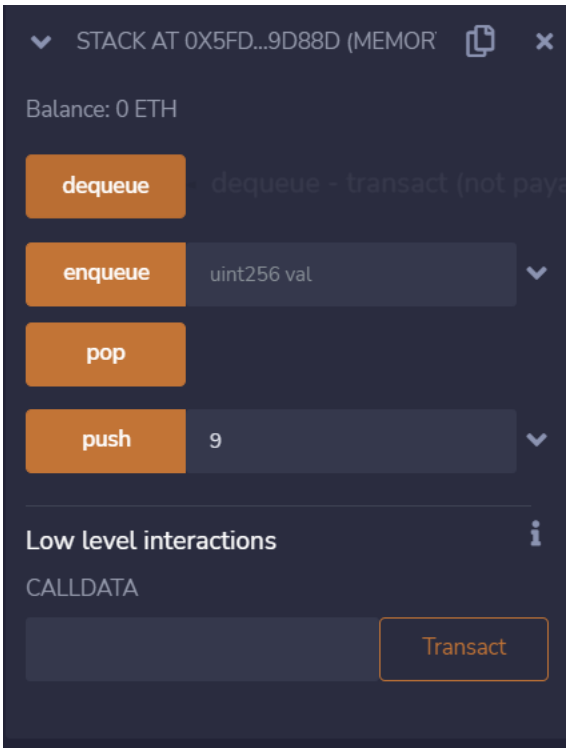
```
    }
    // View function
    function viewFunc() view returns(address) {
        return msg.sender;
    }
    // No return function
    function noReturnFunc() public {
    }
}
```

3. Write Smart contracts to perform STACK and QUEUE operations in solidity.

```
pragma solidity >=0.7.0 <0.9.0;

contract StackQueue {
    uint stackSize;
    uint[] stack;

    function push(uint val) public {
        stack.push(val);
        stackSize++;
    }

    function pop() public returns (uint) {
        uint val = stack[stackSize - 1];
        delete stack[stackSize - 1];
        stackSize--;
        return val;
    }

    uint queueSize;
    uint[] queue;

    function enqueue(uint val) public {
        queue.push(val);
        queueSize++;
    }

    function dequeue() public returns (uint) {
        uint val = queue[0];
        delete queue[0];
        queueSize--;
        return val;
```

Vaneet Singla
102015136
3NC6

```
    }
}
```



Vaneet Singla
102015136
3NC6

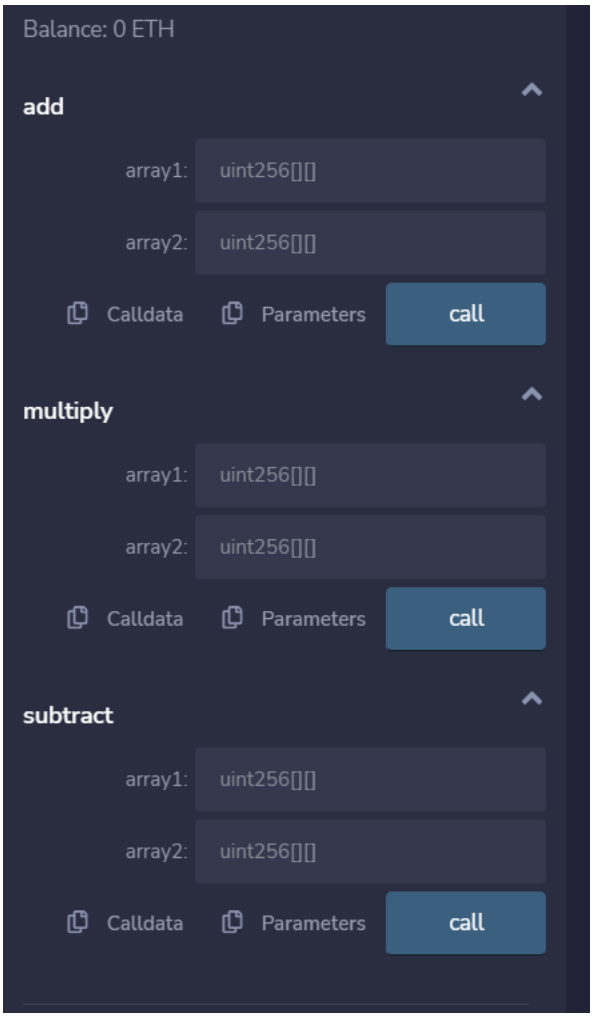
4. Write different contracts in a single file with different functions to perform multidimensional array operations.

```
pragma solidity >=0.7.0 <0.9.0;

contract MultidimensionalArrayOperation {

    // Function to add two multidimensional arrays
    function add(uint[][] memory array1, uint[][] memory array2) public view
returns (uint[][] memory) {
        uint[][] memory res = new uint[][](array1.length);
        for (uint i = 0; i < array1.length; i++) {
            res[i] = new uint[](array1[i].length);
            for (uint j = 0; j < array1[i].length; j++) {
                res[i][j] = array1[i][j] + array2[i][j];
            }
        }
        return res;
    }

    // Function to subtract two multidimensional arrays
    function subtract(uint[][] memory array1, uint[][] memory array2) public view
returns (uint[][] memory) {
        uint[][] memory res = new uint[][](array1.length);
        for (uint i = 0; i < array1.length; i++) {
            res[i] = new uint[](array1[i].length);
            for (uint j = 0; j < array1[i].length; j++) {
```

Vaneet Singla
102015136
3NC6

```
                res[i][j] = array1[i][j] - array2[i][j];
            }
        }
        return res;
    }

    // Function to multiply two multidimensional arrays
    function multiply(uint[][] memory array1, uint[][] memory array2) public view
returns (uint[][] memory) {
        uint[][] memory res = new uint[][](array1.length);
        for (uint i = 0; i < array1.length; i++) {
            res[i] = new uint[](array2[0].length);
            for (uint j = 0; j < array2[0].length; j++) {
                uint sum = 0;
                for (uint k = 0; k < array1[i].length; k++) {
                    sum += array1[i][k] * array2[k][j];
                }
                res[i][j] = sum;
            }
        }
        return res;
    }
}
```

Vaneet Singla
102015136
3NC6



5. Write smart contracts in solidity and call a function from contract1 to contract2 to give input for solving quadratic equation and the computation need to done in function declared in different contract.

```
pragma solidity >=0.7.0 <0.9.0;


contract exp3 {
    int x1;
    int x2;

    function sqrt(int y) internal pure returns (int z) {
        if (y > 3) {
            z = y;
            int x = y / 2 + 1;
```

Vaneet Singla
102015136
3NC6

```
            while (x < z) {
                z = x;
                x = (y / x + x) / 2;
            }
        }
        else if (y != 0) {
            z = 1;
        }
    }

    function f1(int a,int b,int c) public returns(int x1){
        x1=(-b+sqrt((b*b)-(4*a*c)))/(2*a);
        return x1;
    }

    function f2(int a,int b,int c) public returns(int x2){
        x2=(-b-sqrt((b*b)-(4*a*c)))/(2*a);
        return x2;
    }
}

contract ex3{
    exp3 obj=new exp3();
    address vari;

    function f3() public returns(int){
        return obj.f1(1,-2,-15);
    }
    function f4() public returns(int){
        return obj.f2(1,-2,-15);
    }
}
```

Vaneet Singla
102015136
3NC6