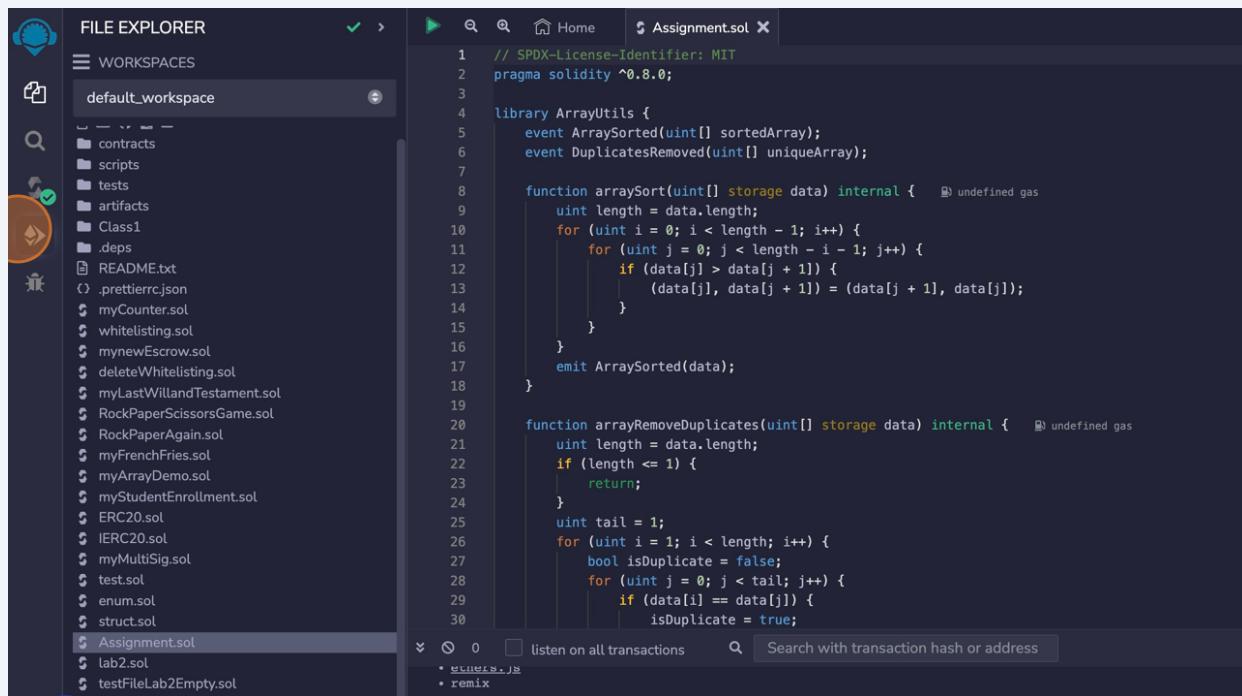


Lab 1

1



The screenshot shows the Scribe IDE interface. On the left is a sidebar with various icons for workspace management, file operations, and project status. The central area has tabs for 'FILE EXPLORER' and 'Assignment.sol'. The 'FILE EXPLORER' tab shows a 'WORKSPACES' section with 'default_workspace' expanded, displaying files like 'contracts', 'scripts', 'tests', 'artifacts', 'Class1', '.deps', 'README.txt', and several Solidity source files such as 'myCounter.sol', 'whitelisting.sol', 'mynewEscrow.sol', 'deleteWhitelisting.sol', 'myLastWillAndTestament.sol', 'RockPaperScissorsGame.sol', 'RockPaperAgain.sol', 'myFrenchFries.sol', 'myArrayDemo.sol', 'myStudentEnrollment.sol', 'ERC20.sol', 'IERC20.sol', 'myMultiSig.sol', 'test.sol', 'enum.sol', 'struct.sol', 'Assignment.sol', 'lab2.sol', and 'testFileLab2Empty.sol'. The 'Assignment.sol' file is currently selected and shown in the main code editor. The code is a Solidity script for array utilities, including functions for sorting arrays and removing duplicates. The right side of the interface includes a search bar and a transaction history section.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

library ArrayUtils {
    event ArraySorted(uint[] sortedArray);
    event DuplicatesRemoved(uint[] uniqueArray);

    function arraySort(uint[] storage data) internal {
        uint length = data.length;
        for (uint i = 0; i < length - 1; i++) {
            for (uint j = 0; j < length - i - 1; j++) {
                if (data[j] > data[j + 1]) {
                    (data[j], data[j + 1]) = (data[j + 1], data[j]);
                }
            }
        }
        emit ArraySorted(data);
    }

    function arrayRemoveDuplicates(uint[] storage data) internal {
        uint length = data.length;
        if (length <= 1) {
            return;
        }
        uint tail = 1;
        for (uint i = 1; i < length; i++) {
            bool isDuplicate = false;
            for (uint j = 0; j < tail; j++) {
                if (data[i] == data[j]) {
                    isDuplicate = true;
                }
            }
            if (!isDuplicate) {
                data[tail] = data[i];
                tail++;
            }
        }
    }
}
```

2

The screenshot shows the Remix IDE interface. On the left, there are configuration fields for GAS LIMIT (3000000), VALUE (0 Wei), and CONTRACT (finalArray - Assignment.sol). Below these are buttons for Deploy (orange) and Publish to IPFS. At the bottom, sections for Transactions recorded and Deployed Contracts are visible. On the right, the Solidity code for the contract is displayed, showing two functions: arraySort and arrayRemoveDuplicates. A circled area highlights the Deploy button.

```

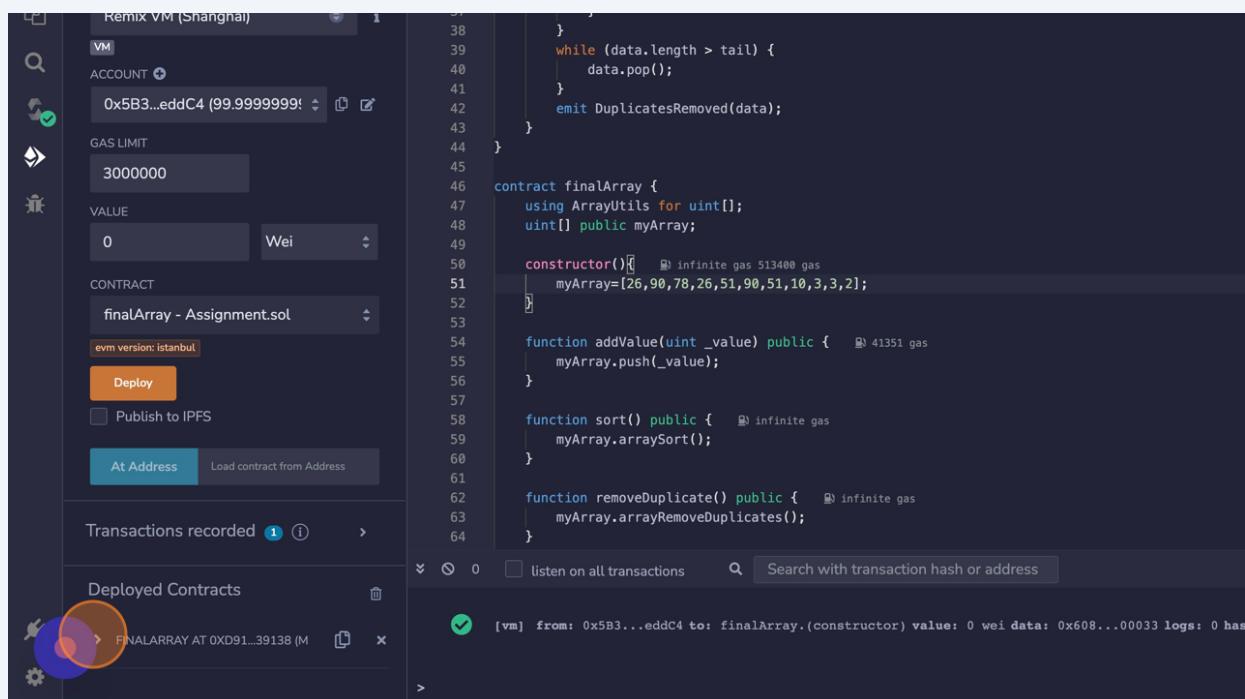
8   function arraySort(uint[] storage data) internal {
9     uint length = data.length;
10    for (uint i = 0; i < length - 1; i++) {
11      for (uint j = 0; j < length - i - 1; j++) {
12        if (data[j] > data[j + 1]) {
13          (data[j], data[j + 1]) = (data[j + 1],
14                                     data[j]);
15      }
16    }
17  }
18  emit ArraySorted(data);
19
20  function arrayRemoveDuplicates(uint[] storage data) internal {
21    uint length = data.length;
22    if (length <= 1) {
23      return;
24    }
25    uint tail = 1;
26    for (uint i = 1; i < length; i++) {
27      bool isDuplicate = false;
28      for (uint j = 0; j < tail; j++) {
29        if (data[i] == data[j]) {
30          isDuplicate = true;
        }
      }
    }
  }

```

3

The screenshot shows the Remix IDE interface after deployment. The left sidebar shows the environment (Remix VM [Shanghai]), account (0x5B3...eddC4), gas limit (3000000), and value (0 Wei). The CONTRACT section shows the deployed contract 'finalArray - Assignment.sol'. The right side displays the Solidity code with a circled area highlighting the constructor. Below the code, a transaction log is shown: '[vm] from: 0x5B3...eddC4 to: finalArray.(constructor) value: 0 wei data: 0x608...00033 logs: 0 hash: 0xae5...8d10c'. A circled checkmark icon is located at the bottom left of the log area.

4



The screenshot shows the Remix VM interface. On the left, there are several icons: a file folder, a magnifying glass, a green checkmark, a diamond, and a gear. Below these are sections for ACCOUNT (set to 0x5B3...eddC4), GAS LIMIT (set to 3000000), and VALUE (set to 0 Wei). The CONTRACT dropdown is set to "finalArray - Assignment.sol". The code editor on the right contains the Solidity code for the finalArray contract. A transaction log at the bottom indicates a successful deployment of the contract to address 0xD91...39138.

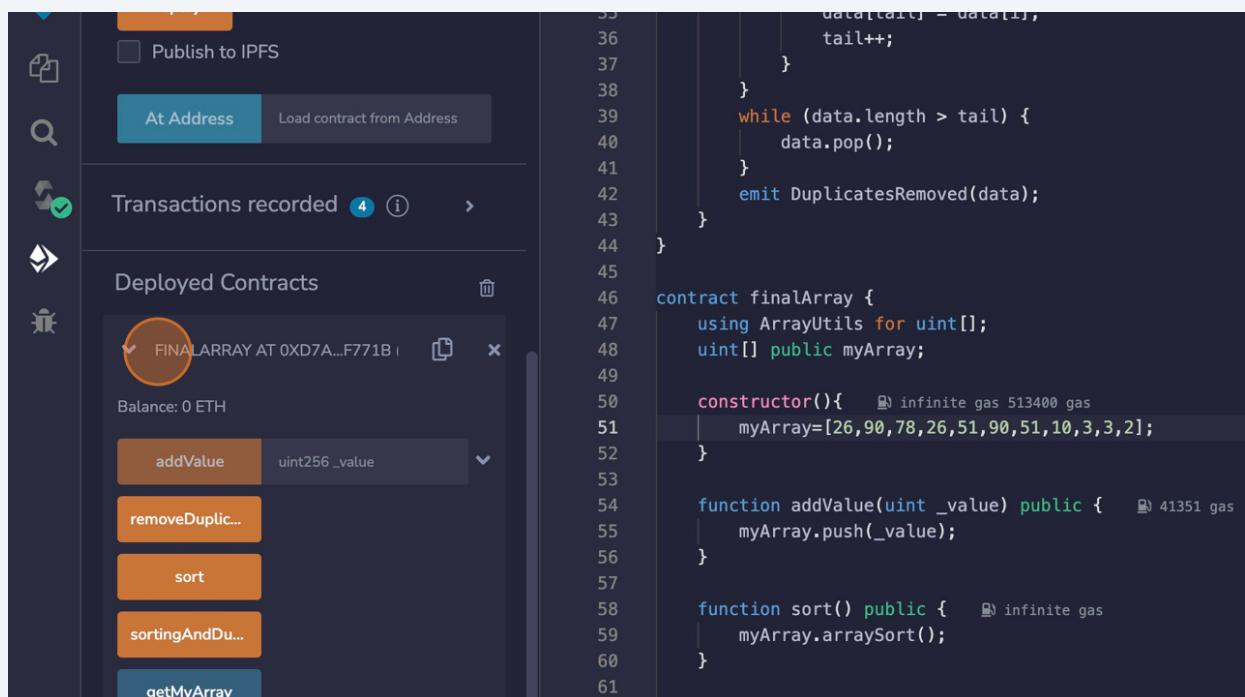
```

Remix VM (Shanghai) 1
VM
ACCOUNT 0x5B3...eddC4 (99.9999999! ✓)
GAS LIMIT 3000000
VALUE 0 Wei
CONTRACT finalArray - Assignment.sol
evm version: istanbul
Deploy
Publish to IPFS
At Address Load contract from Address
Transactions recorded 1 ⓘ >
Deployed Contracts FINALARRAY AT 0xD91...39138 (M) ⚙️ ✎ ✖
[vm] from: 0x5B3...eddC4 to: finalArray.(constructor) value: 0 wei data: 0x608...00033 logs: 0 has

```

5

Double-click "FINALARRAY AT 0XD7A...F771B (MEMORY)"



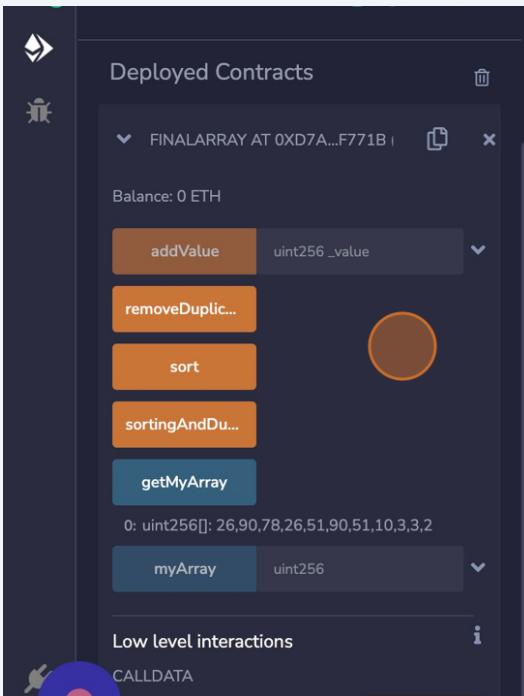
The screenshot shows the Remix VM interface after the finalArray contract has been deployed. The Deployed Contracts section now lists the deployed contract "FINALARRAY AT 0XD7A...F771B". The "At Address" button is highlighted. Below the contract list, there are several buttons: addValue (with input uint256 _value), removeDuplic..., sort, sortingAndDu..., and getMyArray. The code editor on the right shows the same Solidity code as in step 4, with the constructor parameters set to myArray=[26,90,78,26,51,90,51,10,3,3,2].

```

Publish to IPFS
At Address Load contract from Address
Transactions recorded 4 ⓘ >
Deployed Contracts FINALARRAY AT 0XD7A...F771B (M) ⚙️ ✎ ✖
Balance: 0 ETH
addValue uint256 _value
removeDuplic...
sort
sortingAndDu...
getMyArray

```

6 Click here.



```
43 }
44 }
45
46 contract finalArray {
47   using ArrayUtils for uint[];
48   uint[] public myArray;
49
50   constructor() {
51     myArray=[26,90,78,26,51,90,51,10,3,3,2];
52   }
53
54   function addValue(uint _value) public {
55     myArray.push(_value);
56   }
57
58   function sort() public {
59     myArray.arraySort();
60   }
61
62   function removeDuplicate() public {
63     myArray.arrayRemoveDuplicates();
64 }
```

Balance: 0 ETH

addValue uint256 _value

removeDuplicat...

sort

sortingAndDu...

getMyArray

myArray uint256

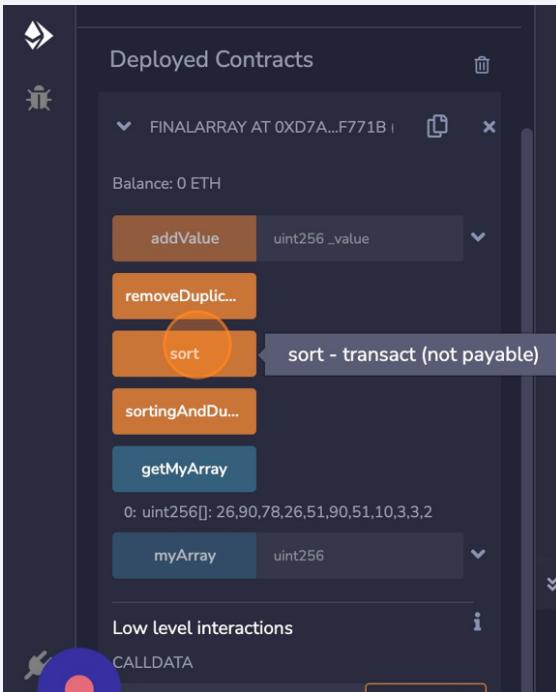
Low level interactions

CALLDATA

0: uint256[]: 26,90,78,26,51,90,51,10,3,3,2

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to 0xd7a...f771b

7 Click here.



```
44 }
45
46 contract finalArray {
47   using ArrayUtils for uint[];
48   uint[] public myArray;
49
50   constructor() {
51     myArray=[26,90,78,26,51,90,51,10,3,3,2];
52   }
53
54   function addValue(uint _value) public {
55     myArray.push(_value);
56   }
57
58   function sort() public {
59     myArray.arraySort();
60   }
61
62   function removeDuplicate() public {
63     myArray.arrayRemoveDuplicates();
64 }
```

Balance: 0 ETH

addValue uint256 _value

removeDuplicat...

sort

sort - transact (not payable)

sortingAndDu...

getMyArray

myArray uint256

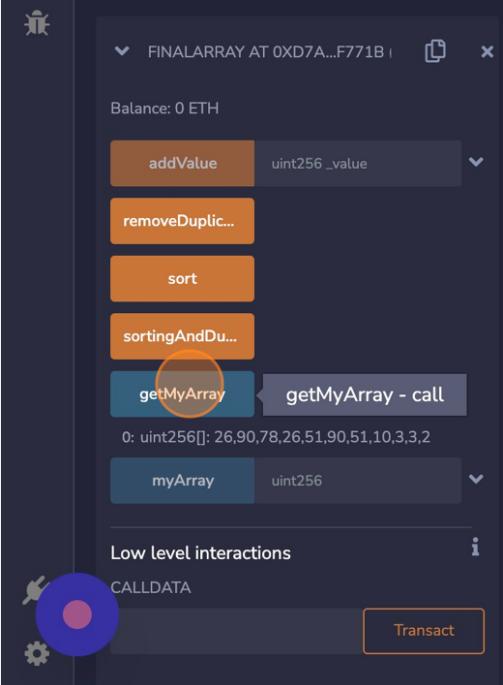
Low level interactions

CALLDATA

0: uint256[]: 26,90,78,26,51,90,51,10,3,3,2

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to 0xd7a...f771b

8 Click here.



```
contract FinalArray {
    using ArrayUtils for uint[];
    uint[] public myArray;

    constructor() {
        myArray=[26,90,78,26,51,90,51,10,3,3,2];
    }

    function addValue(uint _value) public {
        myArray.push(_value);
    }

    function sort() public {
        myArray.arraySort();
    }

    function removeDuplicate() public {
        myArray.arrayRemoveDuplicates();
    }
}
```

Balance: 0 ETH

addValue uint256 _value

removeDuplic...

sort

sortingAndDu...

getMyArray

getMyArray - call

myArray uint256

Low level interactions

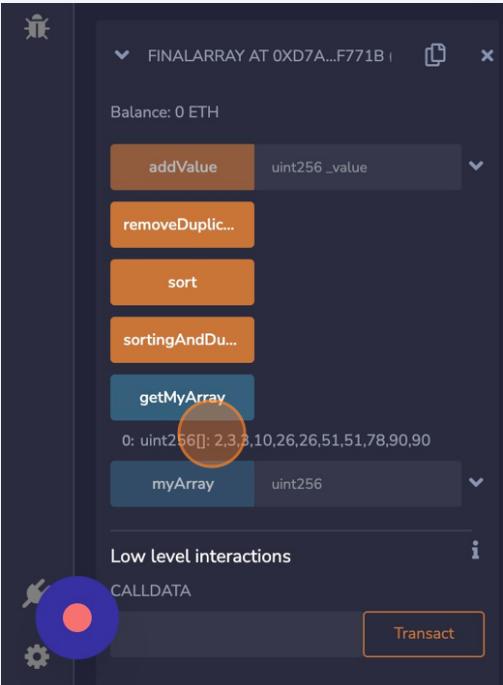
CALLDATA

Transact

0 listen on all transactions Search with transaction h

[vm] from: 0x5B3...eddC4 to: finalArray.sort() 0xD7A...F7

9 It shows "uint256[]: 2,3,3,10,26,26,51,51,78,90,90"



```
contract FinalArray {
    using ArrayUtils for uint[];
    uint[] public myArray;

    constructor() {
        myArray=[26,90,78,26,51,90,51,10,3,3,2];
    }

    function addValue(uint _value) public {
        myArray.push(_value);
    }

    function sort() public {
        myArray.arraySort();
    }

    function removeDuplicate() public {
        myArray.arrayRemoveDuplicates();
    }
}
```

Balance: 0 ETH

addValue uint256 _value

removeDuplic...

sort

sortingAndDu...

getMyArray

0: uint256[]: 2,3,3,10,26,26,51,51,78,90,90

myArray uint256

Low level interactions

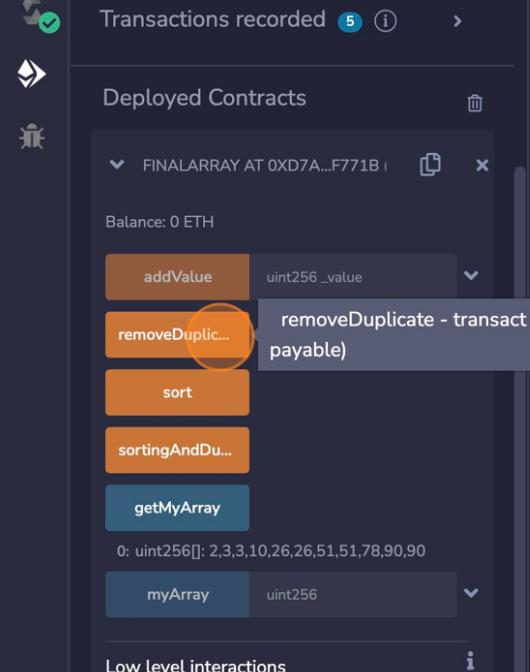
CALLDATA

Transact

0 listen on all transactions Search with transaction h

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 t

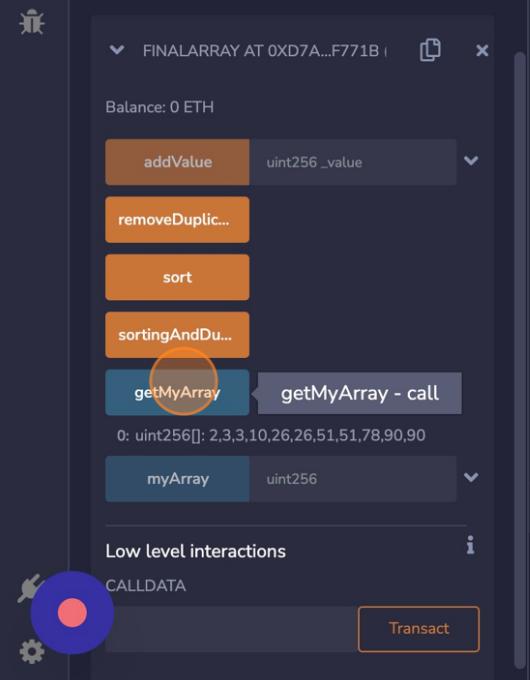
10 Click here.



The screenshot shows the Truffle UI interface. On the left, there's a sidebar with icons for file operations. The main area has tabs for "Transactions recorded" (with 5 items) and "Deployed Contracts". Under "Deployed Contracts", a contract named "FINALARRAY AT 0xD7A...F771B" is listed. Below it, the balance is shown as "0 ETH". A list of functions is provided: "addValue", "removeDuplic...", "sort", "sortingAndDu...", "getMyArray". The "removeDuplic..." button is highlighted with a yellow circle. A tooltip for this button says "removeDuplicate - transact (not payable)". To the right, the Solidity code for the contract is displayed, showing the implementation of the removeDuplicate function. At the bottom, there are buttons for "Low level interactions" and "CALLDATA", and a "Transact" button.

```
42     emit DuplicatesRemoved(data);
43 }
44 }
45
46 contract finalArray {
47     using ArrayUtils for uint[];
48     uint[] public myArray;
49
50     constructor() {
51         myArray=[26,90,78,26,51,90,51,10,3,3,2];
52     }
53
54     function addValue(uint _value) public {
55         myArray.push(_value);
56     }
57
58     function sort() public {
59         myArray.arraySort();
60     }
61
62     function removeDuplicate() public {
63         myArray.arrayRemoveDuplicates();
64     }
}
```

11 Click here.



The screenshot shows the Truffle UI interface after a transaction has been executed. The "removeDuplic..." button is now highlighted with a yellow circle. A tooltip for this button says "getMyArray - call". Below the contract interface, a transaction history entry is visible: "[vm] from: 0x5B3...eddC4 to: finalArray.removeDuplicate() hash: 0x991...4d4b5". The rest of the interface is similar to the previous screenshot, showing the contract code and interaction buttons.

```
40 contract FINALARRAY {
41     using ArrayUtils for uint[];
42     uint[] public myArray;
43
44     constructor() {
45         myArray=[26,90,78,26,51,90,51,10,3,3,2];
46     }
47
48     function addValue(uint _value) public {
49         myArray.push(_value);
50     }
51
52     function sort() public {
53         myArray.arraySort();
54     }
55
56     function removeDuplicate() public {
57         myArray.arrayRemoveDuplicates();
58     }
59
60     function getMyArray() public view returns(uint[]) {
61         return myArray;
62     }
63 }
```

12

The screenshot shows a blockchain development environment. On the left, there is a sidebar with a purple circular icon and a 'CALLDATA' button. Below it are several orange buttons: 'addValue', 'removeDuplic...', 'sort', 'sortingAndDu...', 'getMyArray', and 'myArray'. The 'getMyArray' button is highlighted with a red circle. To its right, a dropdown menu shows the array values: '0: uint256[]: 2,3,10,26,51,78,90'. On the right side, the code editor displays the source code for the 'FINALARRAY' contract:

```
40 contract FINALARRAY {
41     using ArrayUtils for uint[];
42     uint[] public myArray;
43
44     constructor() {
45         myArray=[26,90,78,26,51,90,51,10,3,3,2];
46     }
47
48     function addValue(uint _value) public {
49         myArray.push(_value);
50     }
51
52     function sort() public {
53         myArray.arraySort();
54     }
55
56     function removeDuplicate() public {
57         myArray.arrayRemoveDuplicates();
58     }
59
60 }
```

Below the code editor, there is a transaction history section with a 'CALL' entry and a search bar.

13

This screenshot is similar to the previous one, showing the 'FINALARRAY' interface. The 'getMyArray' button is now highlighted with a red circle. The array values shown are '0: uint256[]: 2,3,10,26,51,78,90'. The code editor on the right shows the same source code for the 'FINALARRAY' contract. The transaction history at the bottom shows a 'CALL' entry from the address 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to the 'FINALARRAY' contract.