

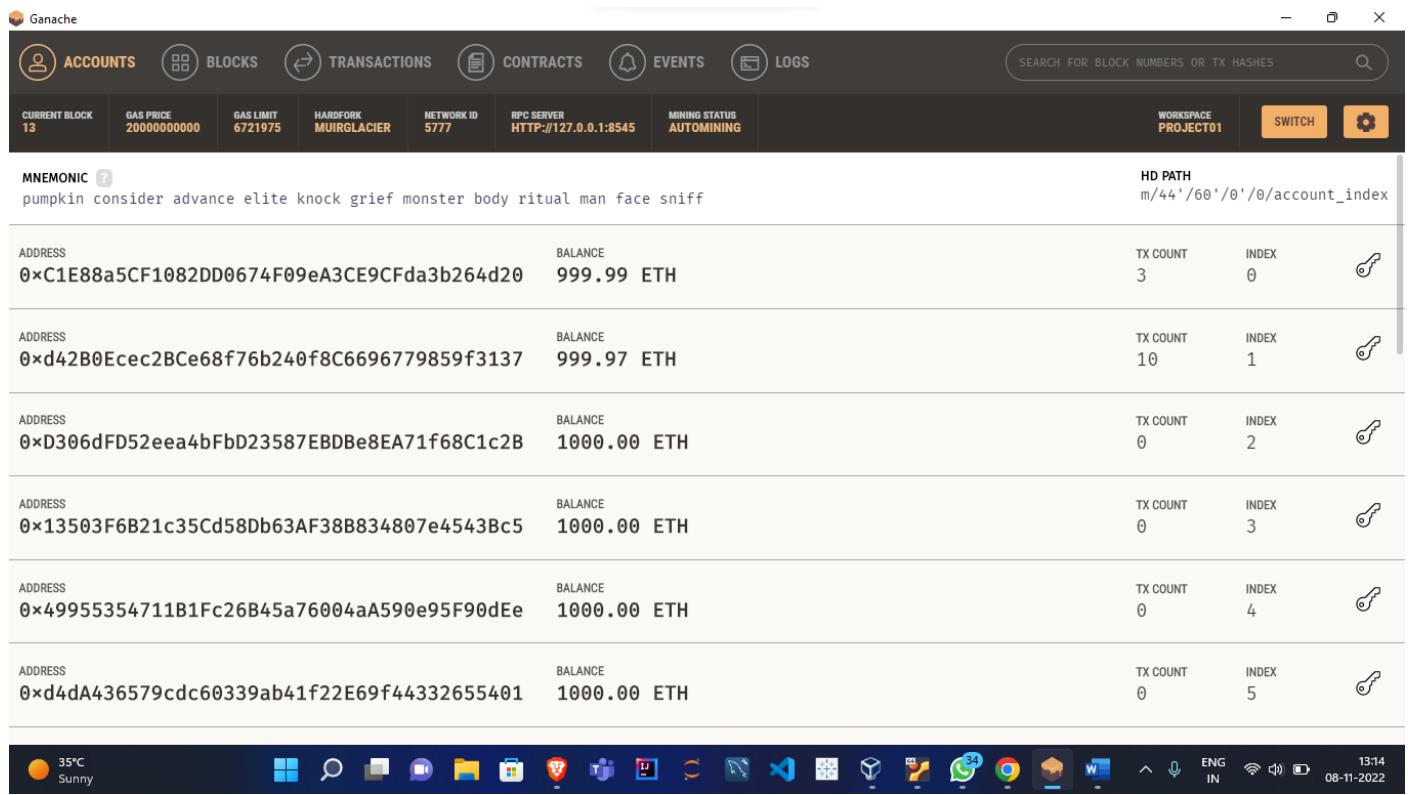
Solidity Beginner Course

Q) COMPLETE SOLIDITY BEGINNER COURSE FROM LEARNETH FROM REMIX IDE. SOLIDITY BEGINNER COURSE SOLUTION CONTRACT SHOULD BE ADDED IN THE FOLLOWING ORDER.

- CHAPTER NAME
- Assignment
- Details of the Assignment
- SOLUTION CONTRACT [ENTIRE SMART CONTRACT CODE]
- Deployed Address on Ganache

1. Introduction:

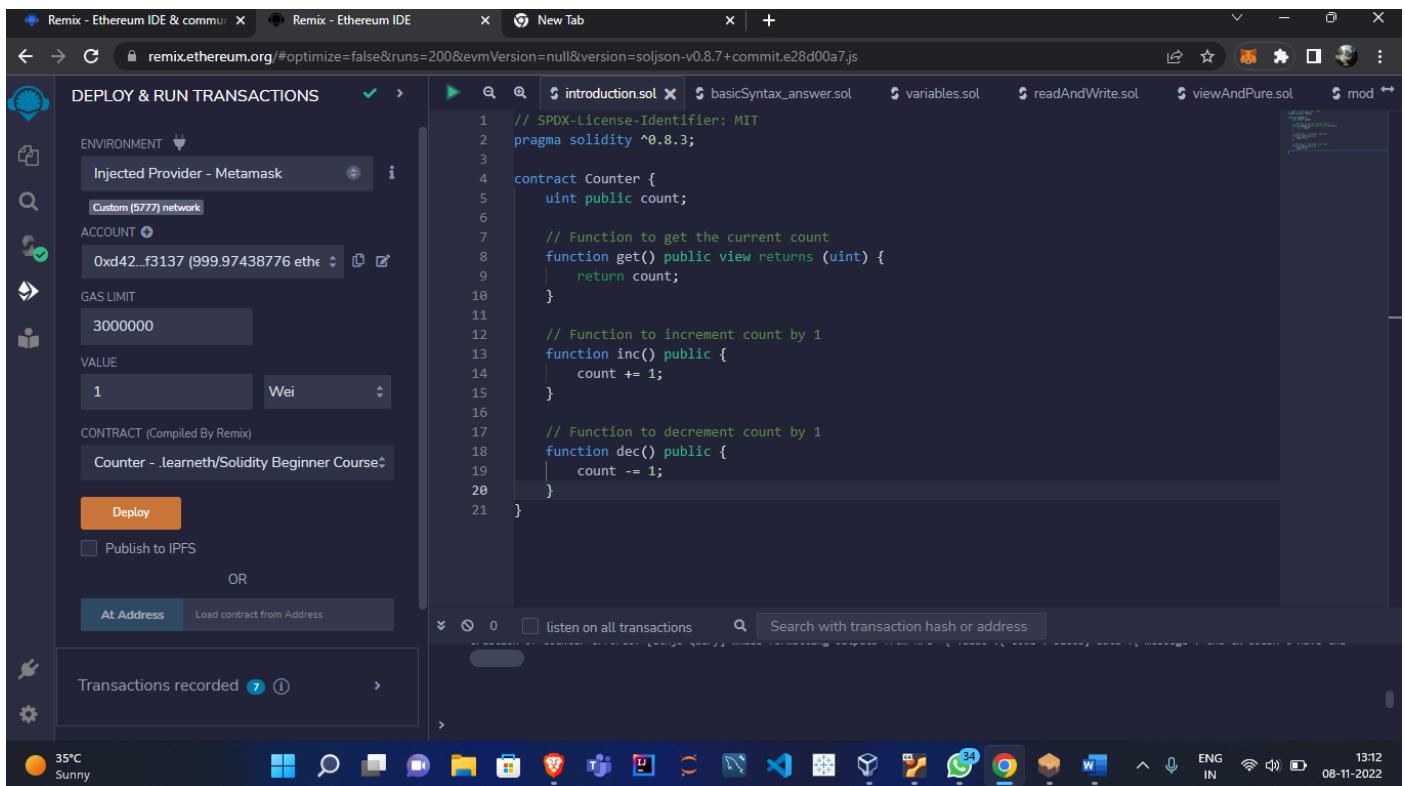
- Compile this contract.
- Deploy it to the JavaScript VM.
- Interact with your contract.



The screenshot shows the Ganache interface with the following account details:

ADDRESS	BALANCE	TX COUNT	INDEX	EDIT
0x1E88a5CF1082DD0674F09eA3CE9CFda3b264d20	999.99 ETH	3	0	🔗
0xd42B0Ecec2BCe68f76b240f8C6696779859f3137	999.97 ETH	10	1	🔗
0xD06dFD52eea4bFbD23587EBDBe8EA71f68C1c2B	1000.00 ETH	0	2	🔗
0x13503F6B21c35Cd58Db63AF38B834807e4543Bc5	1000.00 ETH	0	3	🔗
0x49955354711B1Fc26B45a76004aA590e95F90dEe	1000.00 ETH	0	4	🔗
0xd4dA436579cdc60339ab41f22E69f44332655401	1000.00 ETH	0	5	🔗

At the bottom, the system tray shows a sunny icon, system icons, and the date/time: 13:14 08-11-2022.



Remix - Ethereum IDE & community

Remix - Ethereum IDE

New Tab

remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT

Injected Provider - Metamask

Custom (5777) network

ACCOUNT

0xd42...f3137 (999.97438776 eth)

GAS LIMIT

3000000

VALUE

1 Wei

CONTRACT (Compiled By Remix)

Counter - learneth/Solidity Beginner Course

Deploy

Publish to IPFS

OR

At Address Load contract from Address

Transactions recorded 7

35°C Sunny

Search with transaction hash or address

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;

contract Counter {
    uint public count;

    // Function to get the current count
    function get() public view returns (uint) {
        return count;
    }

    // Function to increment count by 1
    function inc() public {
        count += 1;
    }

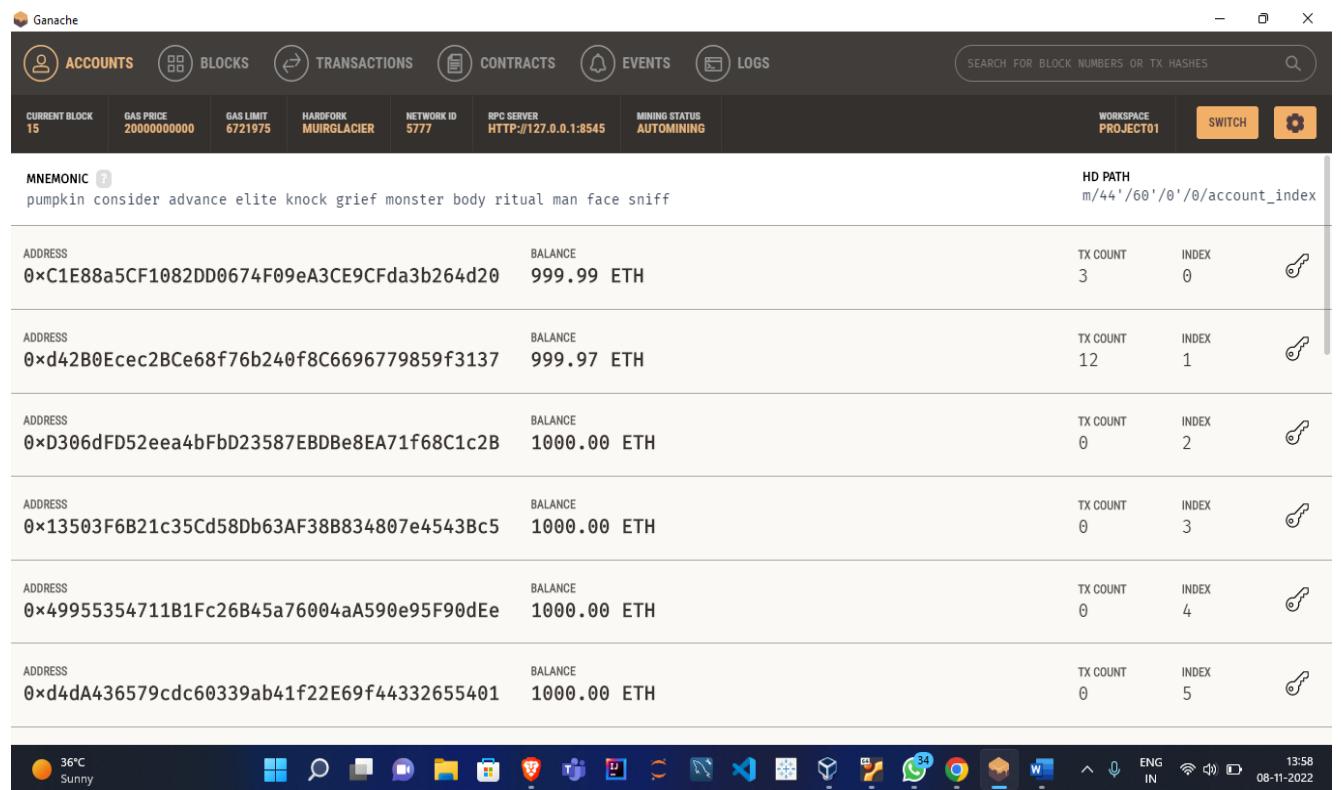
    // Function to decrement count by 1
    function dec() public {
        count -= 1;
    }
}
```

0 listen on all transactions

13:12 08-11-2022

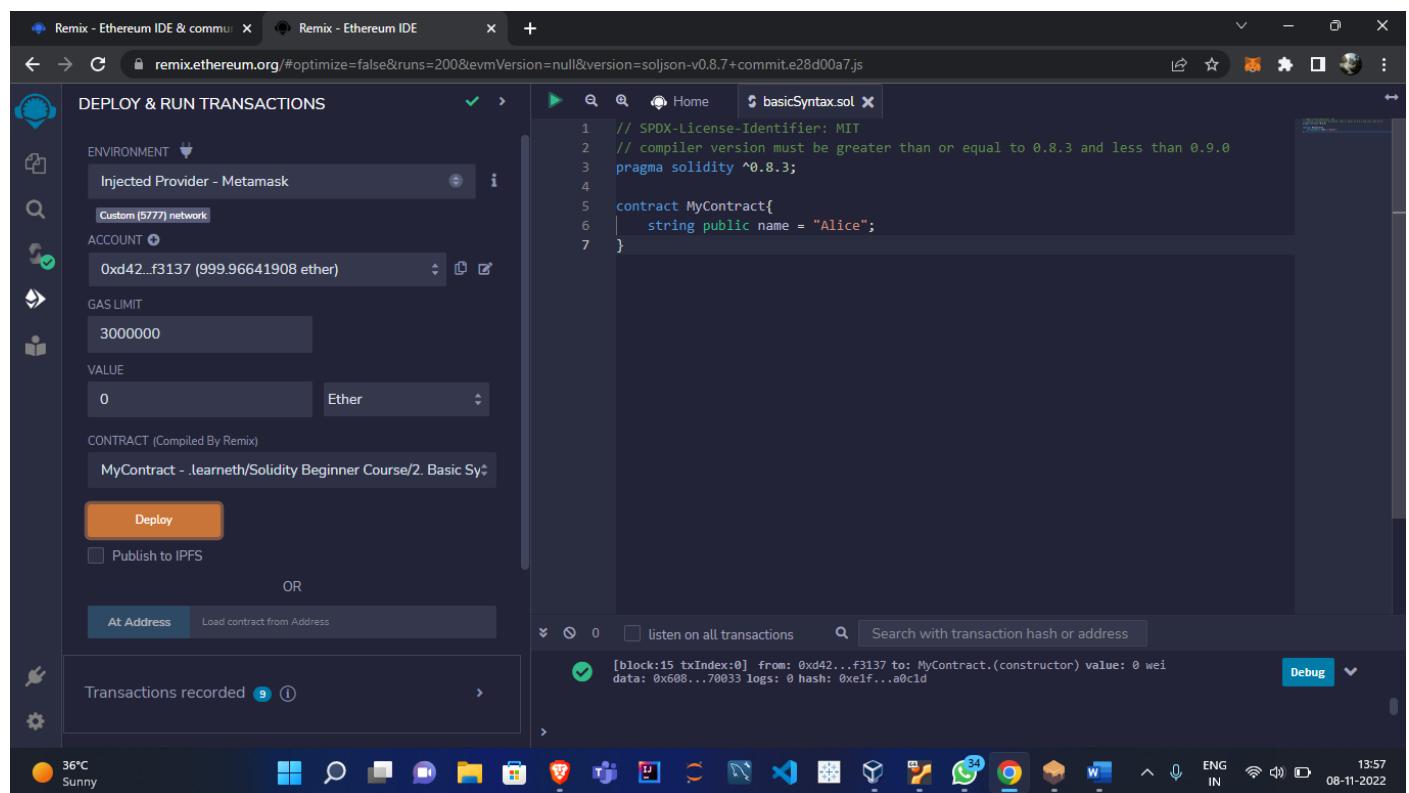
2. Basic Syntax Assignment :

- Delete the HelloWorld contract and its content.
- Create a new contract named "MyContract".
- The contract should have a public state variable called "name" of the type string.
- Assign the value "Alice" to your new variable.



The screenshot shows the Ganache interface. At the top, there are tabs for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, and LOGS. Below the tabs, it shows the current block (15), gas price (2000000000), gas limit (6721975), hardfork (MUIRGLEACIER), network ID (5777), RPC server (HTTP://127.0.0.1:8545), and mining status (AUTOMINING). The workspace is set to PROJECT01. A search bar at the top right allows searching for block numbers or tx hashes. The main area displays a table of accounts. The table has columns for ADDRESS, BALANCE, TX COUNT, and INDEX. Each account row includes a copy icon. The table data is as follows:

ADDRESS	BALANCE	TX COUNT	INDEX
0xC1E88a5CF1082DD0674F09eA3CE9CFda3b264d20	999.99 ETH	3	0
0xd42B0Ecec2BCe68f76b240f8C6696779859f3137	999.97 ETH	12	1
0xD306dFD52eea4bFbD23587EBDBe8EA71f68C1c2B	1000.00 ETH	0	2
0x13503F6B21c35Cd58Db63AF38B834807e4543Bc5	1000.00 ETH	0	3
0x49955354711B1Fc26B45a76004aA590e95F90dEe	1000.00 ETH	0	4
0xd4dA436579cdc60339ab41f22E69f44332655401	1000.00 ETH	0	5



The screenshot shows the Remix Ethereum IDE interface. On the left, the "DEPLOY & RUN TRANSACTIONS" sidebar shows the environment set to "Injected Provider - Metamask" (Custom (5777) network), account "0xd42...f3137 (999.96641908 ether)", gas limit set to 3000000, and value set to 0 Ether. The "CONTRACT (Compiled By Remix)" section shows the "MyContract" contract with the code:

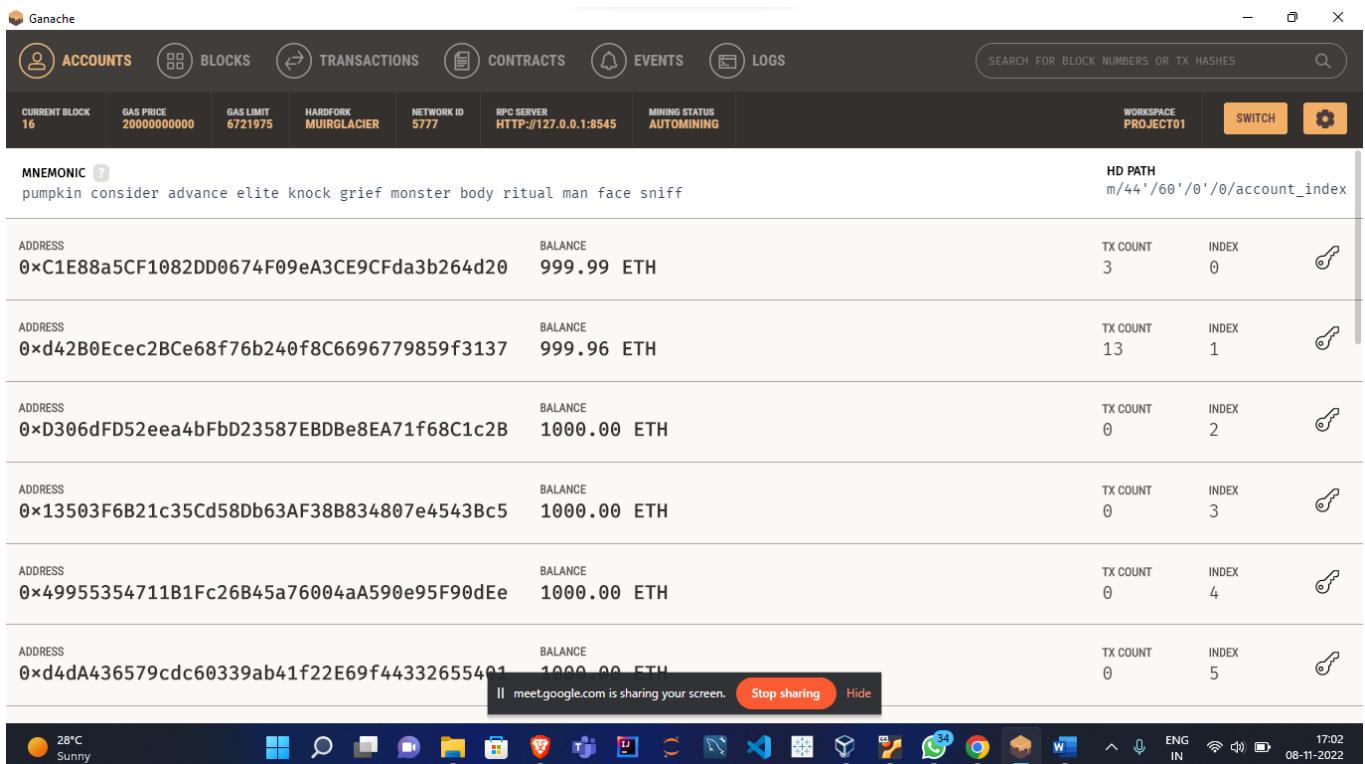
```
// SPDX-License-Identifier: MIT
// compiler version must be greater than or equal to 0.8.3 and less than 0.9.0
pragma solidity ^0.8.3;

contract MyContract{
    string public name = "Alice";
}
```

Below the code, there is a "Deploy" button and a "Publish to IPFS" checkbox. The main workspace shows the "basicSyntax.sol" file with the same code. At the bottom, the transaction details are shown: [block:15 txIndex:0] from: 0xd42...f3137 to: MyContract.(constructor) value: 0 wei data: 0x60...70033 logs: 0 hash: 0xelf...a0cld. The status bar at the bottom indicates it's 13:57 on 08-11-2022.

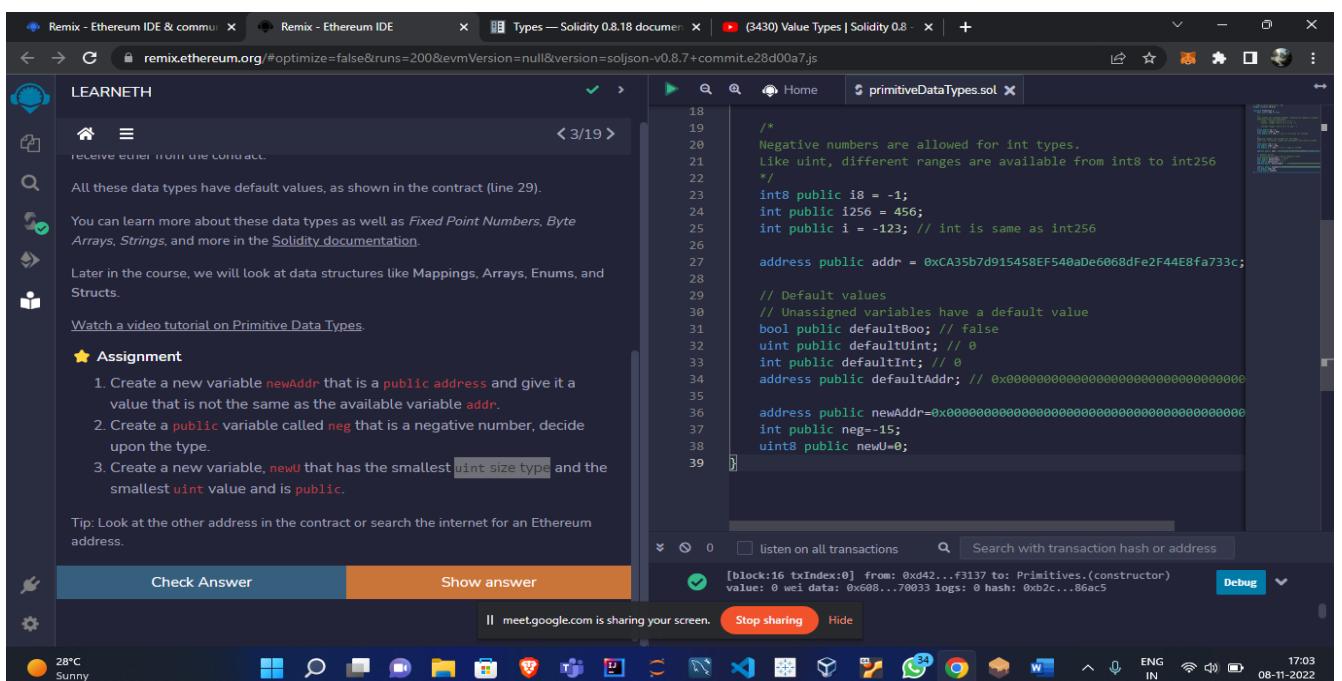
3. Primitive Data Types Assignment:

- Create a new variable `newAddr` that is a public address and give it a value that is not the same as the available variable `addr`.
- Create a public variable called `neg` that is a negative number, decide upon the type.
- Create a new variable, `newU` that has the smallest uint size type and the smallest uint value and is public.



The screenshot shows the Ganache interface with the following details:

- Accounts:** CURRENT BLOCK 16, GAS PRICE 20000000000, GAS LIMIT 6721975, HARDFORK MUIRGLACIER, NETWORK ID 5777, RPC SERVER HTTP://127.0.0.1:8545, MINING STATUS AUTOMINING.
- Contracts:** WORKSPACE PROJECT01, SWITCH, GEAR icon.
- MNEMONIC:** pumpkin consider advance elite knock grief monster body ritual man face sniff.
- HD PATH:** m/44'/60'/0'/0/account_index.
- Accounts List:**
 - ADDRESS 0xC1E88a5CF1082DD0674F09eA3CE9CFda3b264d20 BALANCE 999.99 ETH TX COUNT 3 INDEX 0
 - ADDRESS 0xd42B0Ecec2BCe68f76b240f8C6696779859f3137 BALANCE 999.96 ETH TX COUNT 13 INDEX 1
 - ADDRESS 0xD306dFD52eea4bFbD23587EBDBe8EA71f68C1c2B BALANCE 1000.00 ETH TX COUNT 0 INDEX 2
 - ADDRESS 0x13503F6B21c35Cd58Db63AF38B834807e4543Bc5 BALANCE 1000.00 ETH TX COUNT 0 INDEX 3
 - ADDRESS 0x49955354711B1Fc26B45a76004aA590e95F90dEe BALANCE 1000.00 ETH TX COUNT 0 INDEX 4
 - ADDRESS 0xd4dA436579cdc60339ab41f22E69f44332655401 BALANCE 1000.00 ETH TX COUNT 0 INDEX 5
- Bottom Bar:** 28°C Sunny, various icons (File, Search, Home, etc.), ENG IN, 17:02, 08-11-2022.



The screenshot shows the Remix IDE interface with the following details:

- Remix Tabs:** Remix - Ethereum IDE & commu, Remix - Ethereum IDE, Types — Solidity 0.8.18 document, (3430) Value Types | Solidity 0.8.
- Remix Sidebar (LEARNETH):**
 - Receive ether from the contract.
 - All these data types have default values, as shown in the contract (line 29).
 - You can learn more about these data types as well as *Fixed Point Numbers*, *Byte Arrays*, *Strings*, and more in the [Solidity documentation](#).
 - Later in the course, we will look at data structures like *Mappings*, *Arrays*, *Enums*, and *Structs*.
 - Watch a video tutorial on [Primitive Data Types](#).
 - Assignment:**
 - Create a new variable `newAddr` that is a `public address` and give it a value that is not the same as the available variable `addr`.
 - Create a `public` variable called `neg` that is a negative number, decide upon the type.
 - Create a new variable, `newU` that has the smallest `uint` size type and the smallest `uint` value and is `public`.
 - Tip: Look at the other address in the contract or search the internet for an Ethereum address.
- Code Editor:** primitiveDataTypes.sol (Solidity 0.8.18 document). The code is as follows:

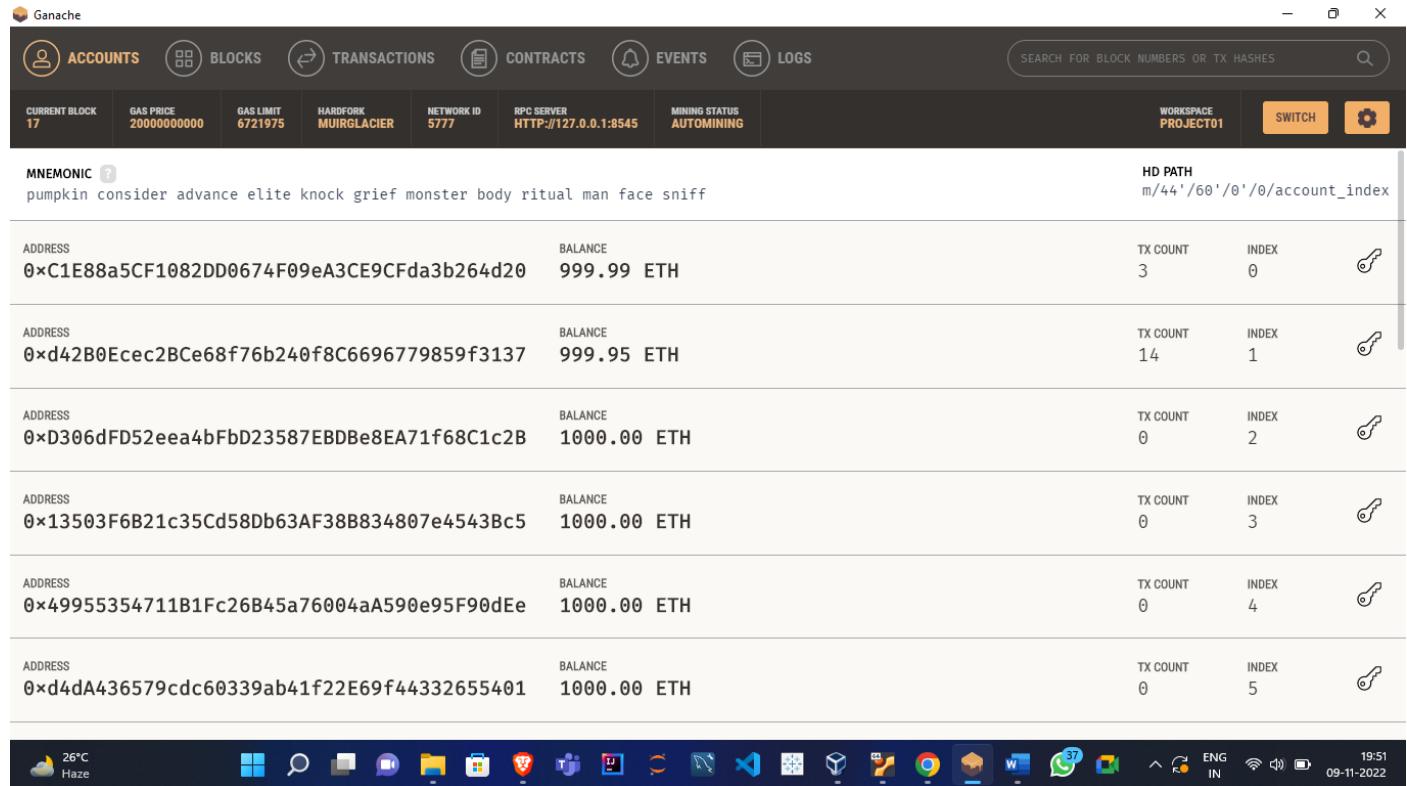
```
18
19  /*
20  * Negative numbers are allowed for int types.
21  * Like uint, different ranges are available from int8 to int256
22  */
23  int8 public i8 = -1;
24  int public i256 = 456;
25  int public i = -123; // int is same as int256
26
27  address public addr = 0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c;
28
29  /*
30  * Unassigned variables have a default value
31  */
32  bool public defaultBool; // false
33  uint public defaultUint; // 0
34  int public defaultInt; // 0
35  address public defaultAddr; // 0x0000000000000000000000000000000000000000
36
37  address public newAddr=0x0000000000000000000000000000000000000000;
38  int public neg=-15;
39  uint8 public newU=0;
```

- Bottom Bar:** 28°C Sunny, various icons (File, Search, Home, etc.), ENG IN, 17:03, 08-11-2022.

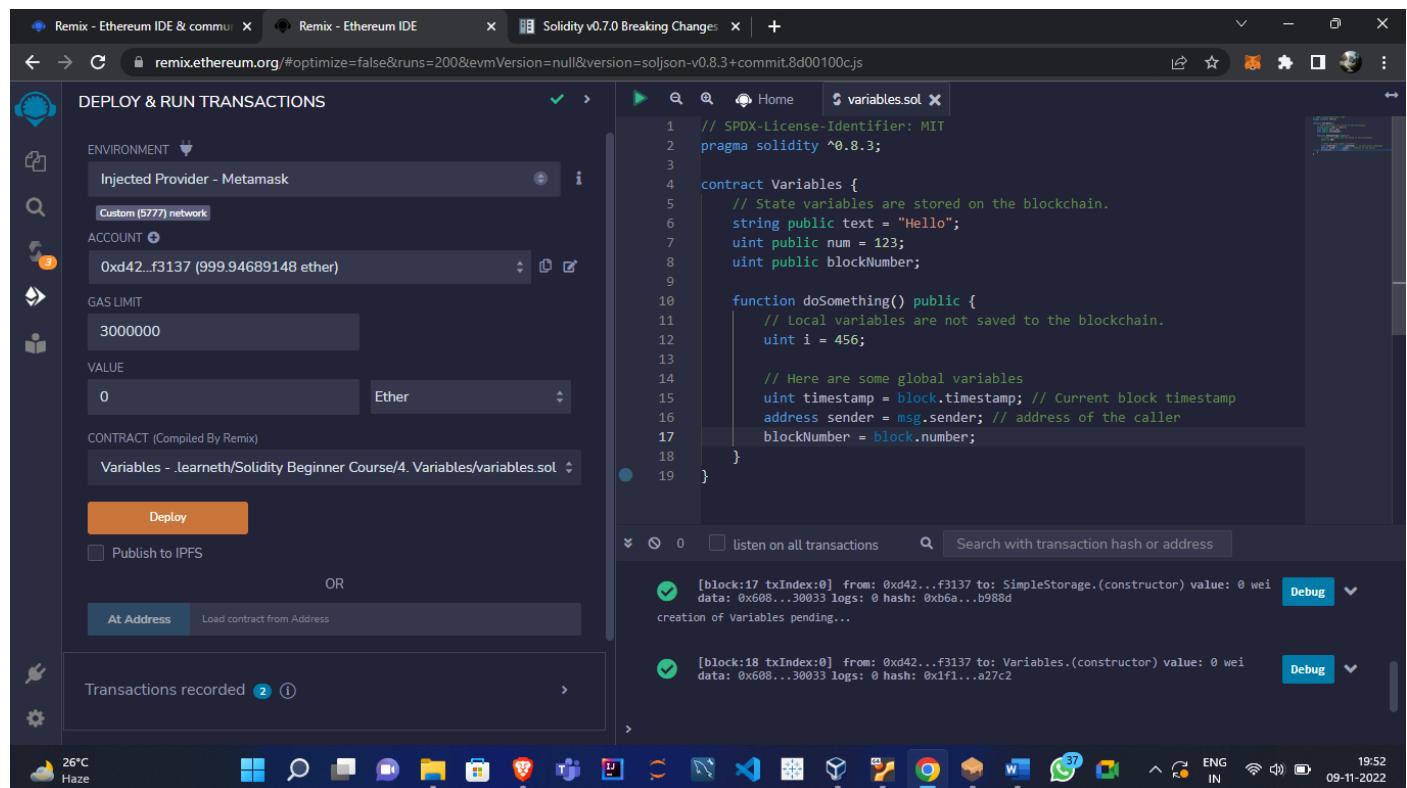
4. Variables : Assignment

Create a new public state variable called `blockNumber`.

Inside the function `doSomething()`, assign the value of the current block number to the state variable `blockNumber`.



The screenshot shows the Ganache UI interface. At the top, there are tabs for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, and LOGS. Below these are network settings: CURRENT BLOCK (17), GAS PRICE (20000000000), GAS LIMIT (6721975), HARDFORK (MUIRGLACIER), NETWORK ID (5777), RPC SERVER (HTTP://127.0.0.1:8545), and MINING STATUS (AUTOMINING). A workspace named PROJECT01 is selected. A search bar at the top right allows searching for block numbers or transaction hashes. The main area displays account details, including a mnemonic seed: "pumpkin consider advance elite knock grief monster body ritual man face sniff". The HD PATH is listed as m/44'/60'/0'/0/account_index. Below this, a table lists five accounts with their addresses, balances (999.99, 999.95, 1000.00, 1000.00, 1000.00 ETH), transaction counts (3, 14, 0, 0, 0), and indices (0, 1, 2, 3, 4). Each account row has a copy icon. The bottom of the screen shows a taskbar with various icons and the date/time 09-11-2022.



The screenshot shows the Remix IDE interface. On the left, the "DEPLOY & RUN TRANSACTIONS" sidebar shows an environment set to "Injected Provider - Metamask" (Custom (5777) network), account 0xd42...f3137 (999.94689148 ether), gas limit 3000000, and value 0 Ether. Below this is a "CONTRACT (Compiled By Remix)" section for "Variables - .learneth/Solidity Beginner Course/4. Variables/variables.sol". The code editor on the right contains the following Solidity code:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;

contract Variables {
    // State variables are stored on the blockchain.
    string public text = "Hello";
    uint public num = 123;
    uint public blockNumber;

    function doSomething() public {
        // Local variables are not saved to the blockchain.
        uint i = 456;

        // Here are some global variables
        uint timestamp = block.timestamp; // Current block timestamp
        address sender = msg.sender; // address of the caller
        blockNumber = block.number;
    }
}
```

At the bottom, the transaction history shows two entries:

- [block:17 txIndex:0] from: 0xd42...f3137 to: SimpleStorage.(constructor) value: 0 wei data: 0x608...30033 logs: 0 hash: 0xb6a...b988d creation of Variables pending...
- [block:18 txIndex:0] from: 0xd42...f3137 to: Variables.(constructor) value: 0 wei data: 0x608...30033 logs: 0 hash: 0x1f1...a27c2

The bottom of the screen shows a taskbar with various icons and the date/time 09-11-2022.

5.1) Functions - Reading and Writing to a State Variable

Ganache

ACCOUNTS BLOCKS TRANSACTIONS CONTRACTS EVENTS LOGS

SEARCH FOR BLOCK NUMBERS OR TX HASHES

CURRENT BLOCK 17 GAS PRICE 20000000000 GAS LIMIT 6721975 HARDFORK MUIRGLACIER NETWORK ID 5777 RPC SERVER HTTP://127.0.0.1:8545 MINING STATUS AUTOMINING

WORKSPACE PROJECT01 SWITCH

MNEMONIC: pumpkin consider advance elite knock grief monster body ritual man face sniff

HD PATH: m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0xC1E88a5CF1082DD0674F09eA3CE9CFda3b264d20	999.99 ETH	3	0
0xd42B0Ecec2BCe68f76b240f8C6696779859f3137	999.95 ETH	14	1
0xD306dFD52eea4bFbD23587EBDBe8EA71f68C1c2B	1000.00 ETH	0	2
0x13503F6B21c35Cd58Db63AF38B834807e4543Bc5	1000.00 ETH	0	3
0x49955354711B1Fc26B45a76004aA590e95F90dEe	1000.00 ETH	0	4
0xd4dA436579cdc60339ab41f22E69f44332655401	1000.00 ETH	0	5

23°C Clear

Remix - Ethereum IDE & community Remix - Ethereum IDE Cheatsheet — Solidity 0.8.18 doc solidity - Compile error - Warning

remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.3+commit.8d00100cjs

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Injected Provider - Metamask

ACCOUNT: 0xd42...f3137 (999.95212308 ether)

GAS LIMIT: 3000000

VALUE: 0 Ether

CONTRACT (Compiled By Remix): SimpleStorage - learneth/Solidity Beginner Course/5.1 Functions - Read

Deploy

Publish to IPFS

OR

At Address: Load contract from Address

Transactions recorded: 1

3 contract SimpleStorage {
4 // State variable to store a number
5 uint public num;
6 bool public b = true;
7
8 // You need to send a transaction to write to a state variable.
9 function set(uint _num) public {
10 | num = _num;
11 }
12
13 // You can read from a state variable without sending a transaction.
14 function get() public view returns (uint) {
15 | return num;
16 }
17
18 function get_b() public view returns (bool) {
19 | return b;
20 }
21 }
22

0 listen on all transactions Search with transaction hash or address

remix

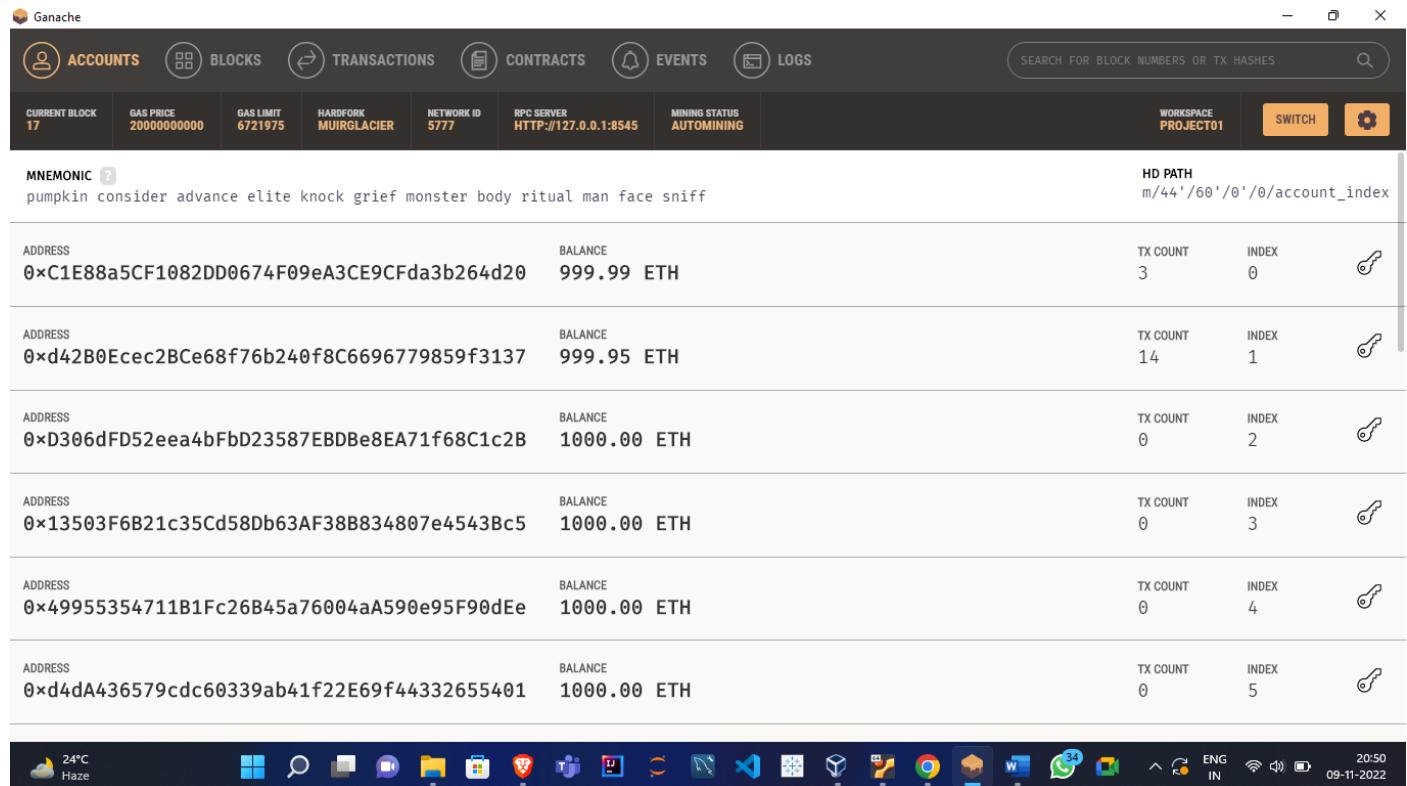
Type the library name to see available commands.
creation of SimpleStorage pending...

[block:17 txIndex:0] from: 0xd42...f3137 to: SimpleStorage.(constructor) value: 0 wei data: 0x608...30033 logs: 0 hash: 0xb6a...b988d Debug

23°C Clear

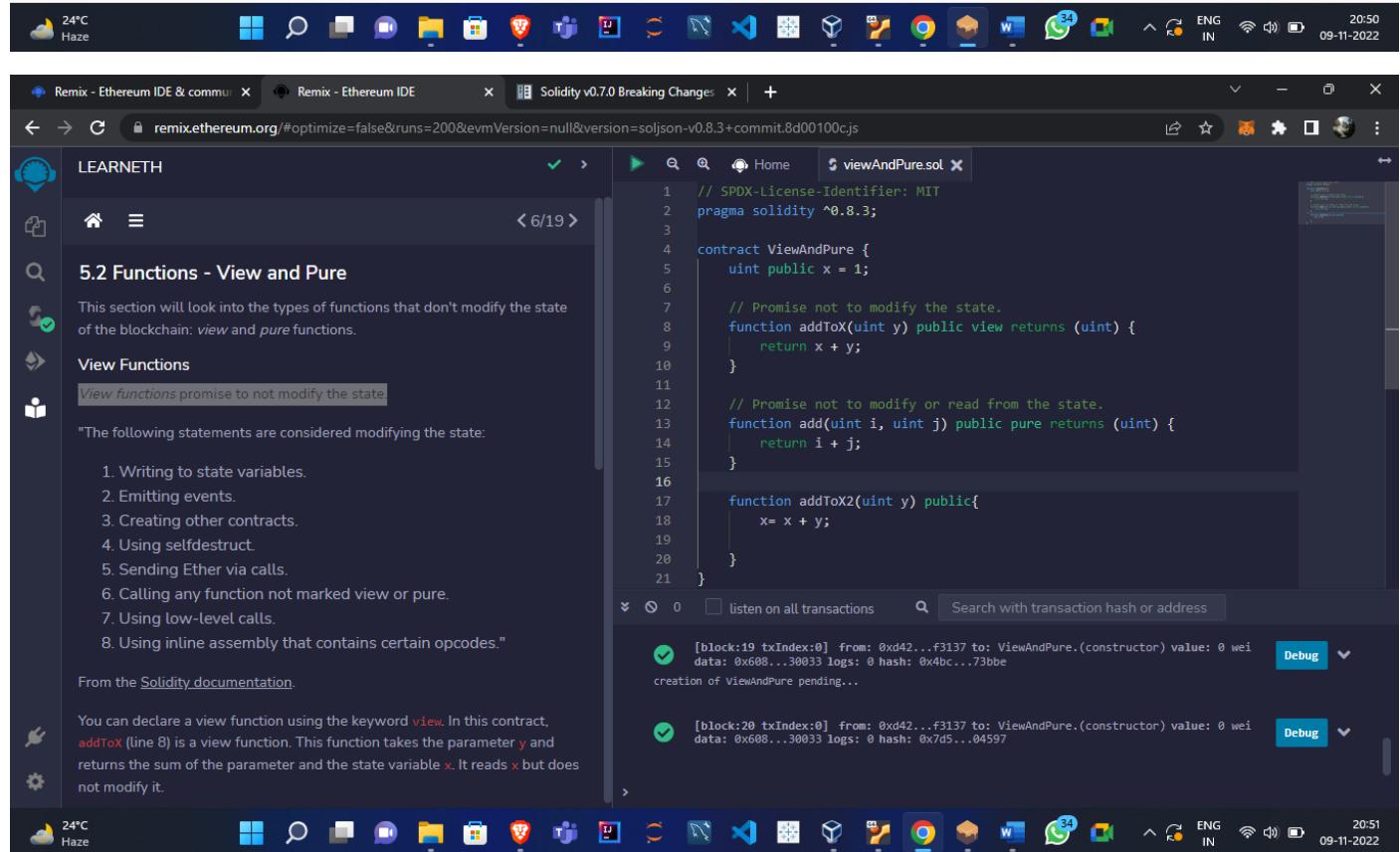
5.2) Functions - View and Pure : Assignment

Create a function called `addToX2` that takes the parameter `y` and updates the state variable `x` with the sum of the parameter and the state variable `x`.



The screenshot shows the Ganache interface with the following details:

- Accounts:** CURRENT BLOCK 17, GAS PRICE 20000000000, GAS LIMIT 6721975, HARDFORK MUIRGLACIER, NETWORK ID 5777, RPC SERVER HTTP://127.0.0.1:8545, MINING STATUS AUTOMINING.
- Mnemonic:** pumpkin consider advance elite knock grief monster body ritual man face sniff
- HD PATH:** m/44'/60'/0'/0/account_index
- Accounts List:** A table showing 6 accounts with their addresses, balances (999.99 to 1000.00 ETH), transaction counts (0 to 14), and indices (0 to 5). Each account has a copy icon.



The screenshot shows the Remix IDE interface with the following details:

- Remix Version:** Solidity v0.7.0 Breaking Changes
- Contract Name:** ViewAndPure
- Code:**

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;

contract ViewAndPure {
    uint public x = 1;

    // Promise not to modify the state.
    function addToX(uint y) public view returns (uint) {
        return x + y;
    }

    // Promise not to modify or read from the state.
    function add(uint i, uint j) public pure returns (uint) {
        return i + j;
    }

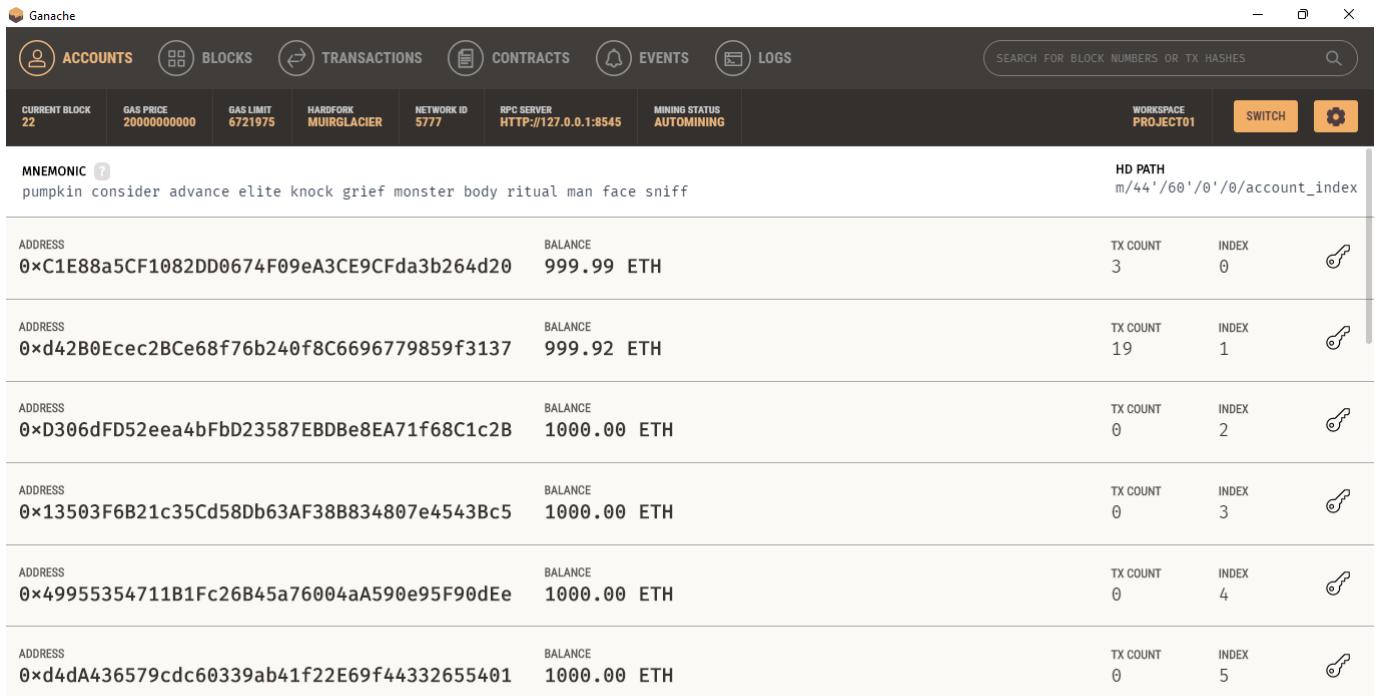
    function addToX2(uint y) public{
        x = x + y;
    }
}
```
- Logs:** Two transaction logs are shown, both from the constructor of the ViewAndPure contract.
- Environment:** Shows a Windows taskbar at the bottom with various application icons.

5.3 Functions - Modifiers and Constructors: Assignment

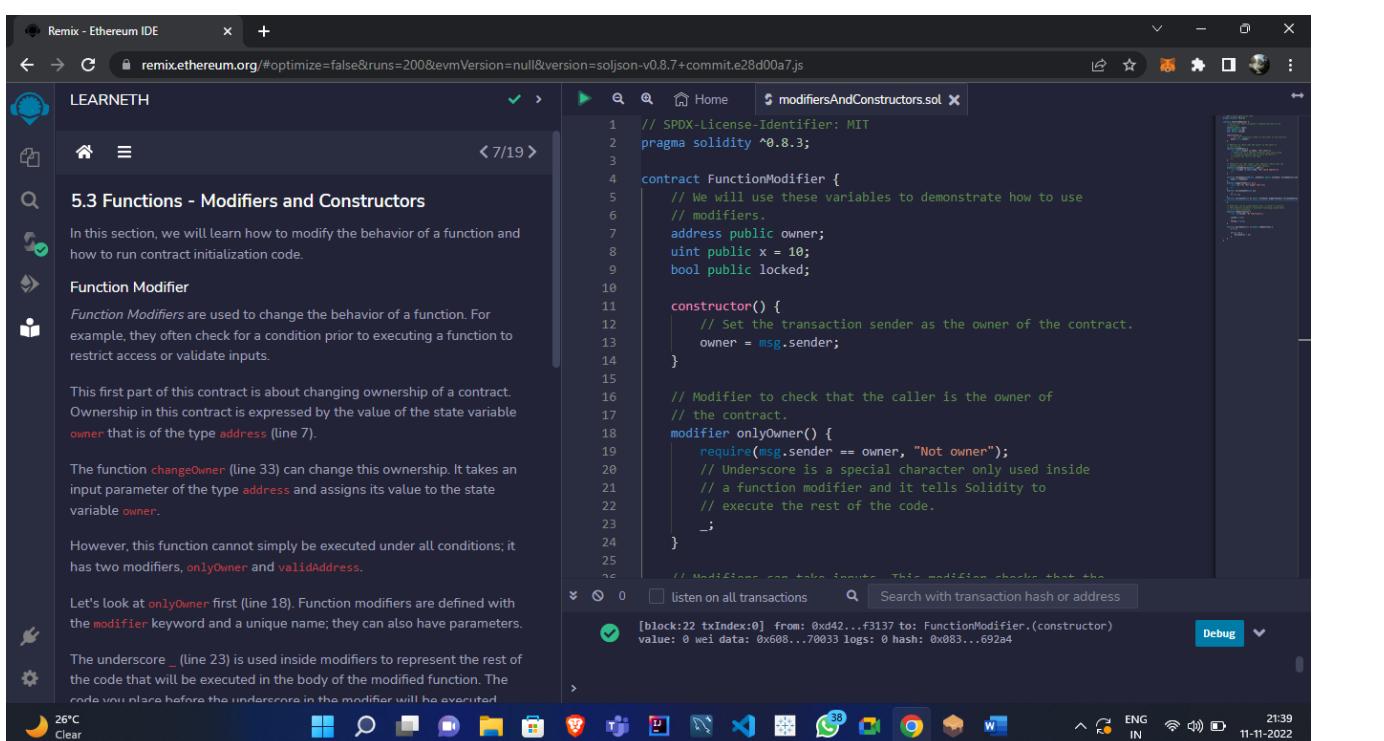
Create a new function, `increaseX` in the contract. The function should take an input parameter of type `uint` and increase the value of the variable `x` by the value of the input parameter.

Make sure that `x` can only be increased.

The body of the function `increaseX` should be empty.



ADDRESS	BALANCE	TX COUNT	INDEX	
0xC1E88a5CF1082DD0674F09eA3CE9CFda3b264d20	999.99 ETH	3	0	
0xd42B0Ecec2BCe68f76b240f8C6696779859f3137	999.92 ETH	19	1	
0xD306dFD52eea4bFbD23587EBDBe8EA71f68C1c2B	1000.00 ETH	0	2	
0x13503F6B21c35Cd58Db63AF38B834807e4543Bc5	1000.00 ETH	0	3	
0x49955354711B1Fc26B45a76004aA590e95F90dEe	1000.00 ETH	0	4	
0xd4dA436579cdc60339ab41f22E69f44332655401	1000.00 ETH	0	5	



LEARNEETH

5.3 Functions - Modifiers and Constructors

In this section, we will learn how to modify the behavior of a function and how to run contract initialization code.

Function Modifier

Function Modifiers are used to change the behavior of a function. For example, they often check for a condition prior to executing a function to restrict access or validate inputs.

This first part of this contract is about changing ownership of a contract. Ownership in this contract is expressed by the value of the state variable `owner` that is of the type `address` (line 7).

The function `changeOwner` (line 33) can change this ownership. It takes an input parameter of the type `address` and assigns its value to the state variable `owner`.

However, this function cannot simply be executed under all conditions; it has two modifiers, `onlyOwner` and `validAddress`.

Let's look at `onlyOwner` first (line 18). Function modifiers are defined with the `modifier` keyword and a unique name; they can also have parameters.

The underscore `_` (line 23) is used inside modifiers to represent the rest of the code that will be executed in the body of the modified function. The code you place before the underscore in the modifier will be executed

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;

contract FunctionModifier {
    // We will use these variables to demonstrate how to use
    // modifiers.
    address public owner;
    uint public x = 10;
    bool public locked;

    constructor() {
        // Set the transaction sender as the owner of the contract.
        owner = msg.sender;
    }

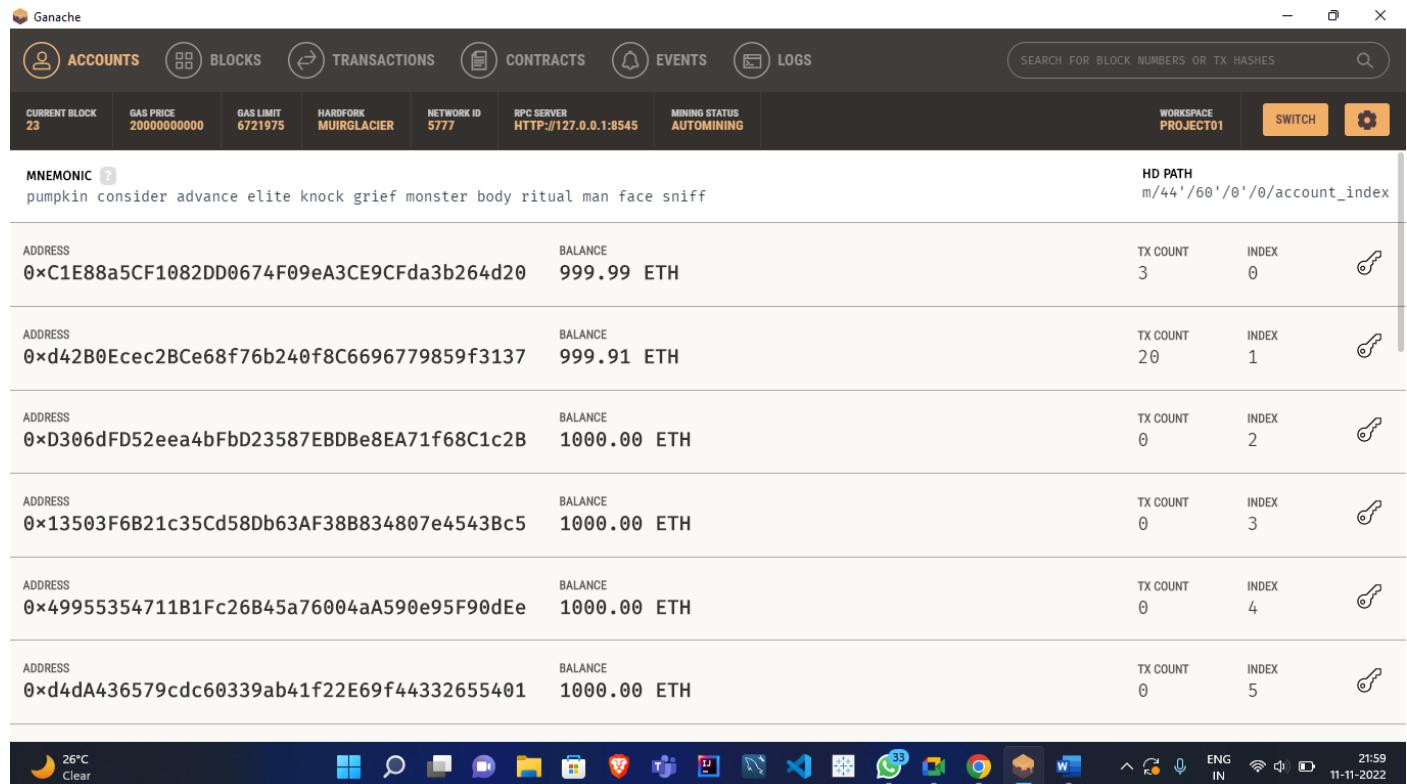
    // Modifier to check that the caller is the owner of
    // the contract.
    modifier onlyOwner() {
        require(msg.sender == owner, "Not owner");
        // Underscore is a special character only used inside
        // a function modifier and it tells Solidity to
        // execute the rest of the code.
    }

    // Modifiers can take function this function shows that the
    // modifier can take function
}
```

[block:22 txIndex:0] from: 0xd42...f3137 to: FunctionModifier.(constructor) value: 0 wei data: 0x608...70033 logs: 0 hash: 0x083...692a4

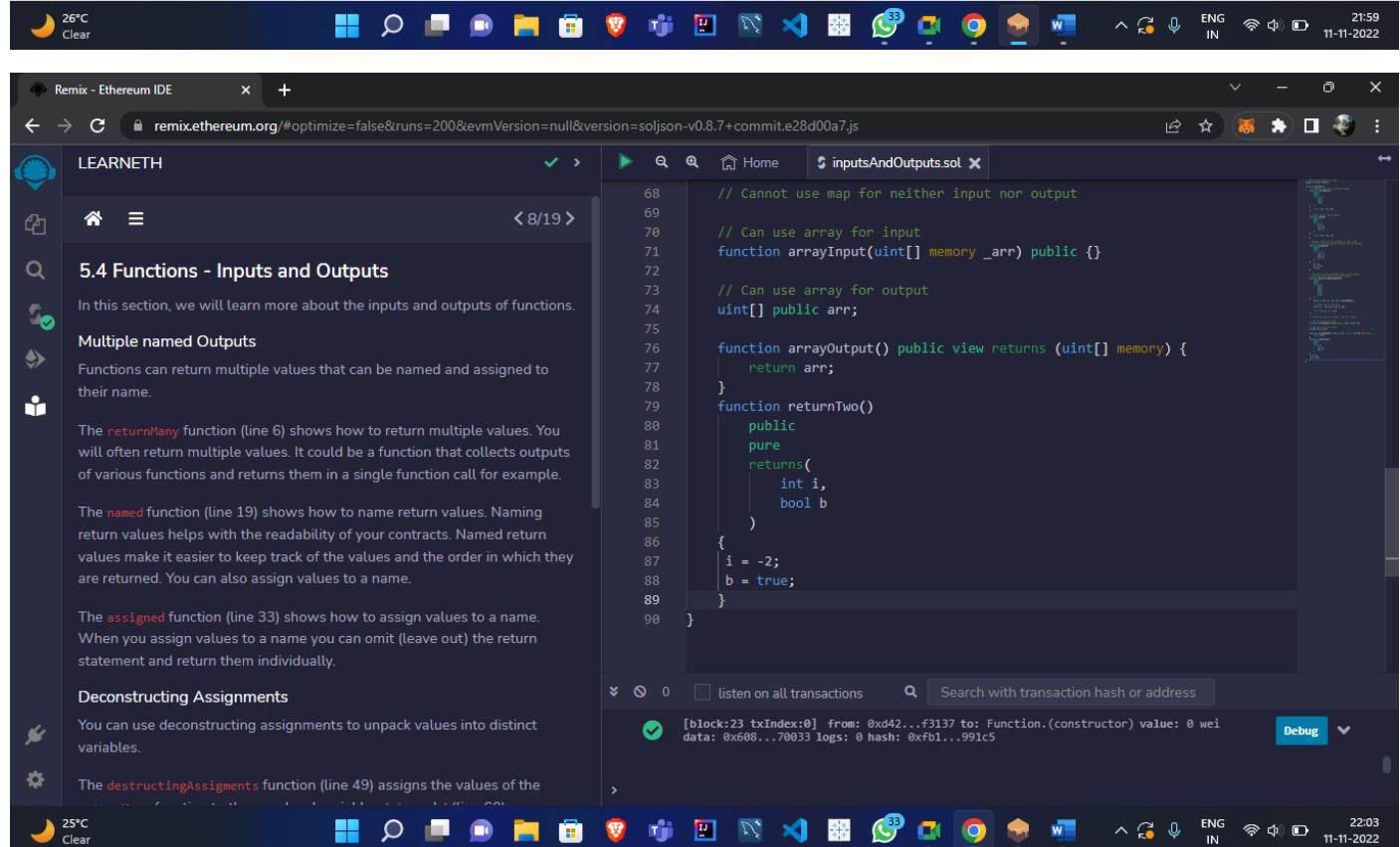
5.4 Functions - Inputs and Outputs :Assignment

Create a new function called `returnTwo` that returns the values -2 and true without using a return statement.



The screenshot shows the Ganache UI interface. At the top, there are tabs for Accounts, Blocks, Transactions, Contracts, Events, and Logs. Below the tabs, there are several status indicators: Current Block (23), Gas Price (20000000000), Gas Limit (6721975), Hardfork (MUIRGLAGIER), Network ID (5777), RPC Server (HTTP://127.0.0.1:8545), and Mining Status (AUTOMINING). A search bar at the top right allows searching for block numbers or tx hashes. A workspace switcher shows 'PROJECT01' and a gear icon for settings. The main table lists accounts with their addresses, balances, transaction counts, and indices. The mnemonic seed 'pumpkin consider advance elite knock grief monster body ritual man face sniff' is shown, along with its HD Path: m/44'/60'/0'/0/account_index.

ADDRESS	BALANCE	TX COUNT	INDEX	
0xC1E88a5CF1082DD0674F09eA3CE9CFda3b264d20	999.99 ETH	3	0	🔑
0xd42B0Ecec2BCe68f76b240f8C6696779859f3137	999.91 ETH	20	1	🔑
0xD306dFD52eea4bFdD23587EBDBe8EA71f68C1c2B	1000.00 ETH	0	2	🔑
0x13503F6B21c35Cd58Db63AF38B834807e4543Bc5	1000.00 ETH	0	3	🔑
0x49955354711B1Fc26B45a76004aA590e95F90dEe	1000.00 ETH	0	4	🔑
0xd4dA436579cdc60339ab41f22E69f44332655401	1000.00 ETH	0	5	🔑



The screenshot shows the Remix - Ethereum IDE interface. The left sidebar contains a navigation menu with sections like LEARNETH, 5.4 Functions - Inputs and Outputs, Multiple named Outputs, Deconstructing Assignments, and The destructingAssignments function. The main area is a code editor with Solidity code. The code includes comments explaining the use of arrays for inputs and outputs, and a function `returnTwo` that returns -2 and true. The Remix interface includes a terminal at the bottom for running commands like `geth` and `ganache`. The status bar at the bottom shows the date and time (11-11-2022, 21:59).

```
// Cannot use map for neither input nor output
// Can use array for input
function arrayInput(uint[] memory _arr) public {}

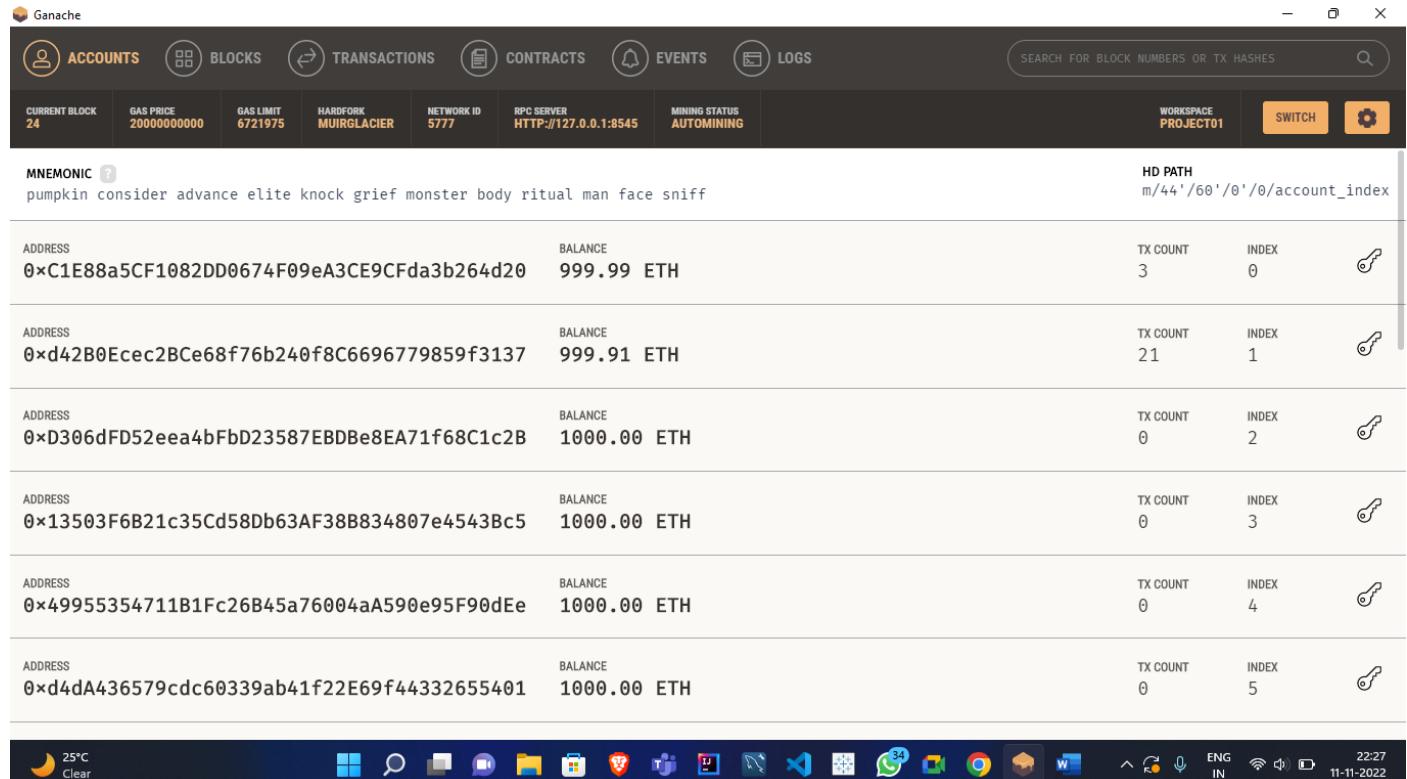
// Can use array for output
uint[] public arr;

function arrayOutput() public view returns (uint[] memory) {
    return arr;
}

function returnTwo()
public
pure
returns(
    int i,
    bool b
)
{
    i = -2;
    b = true;
}
```

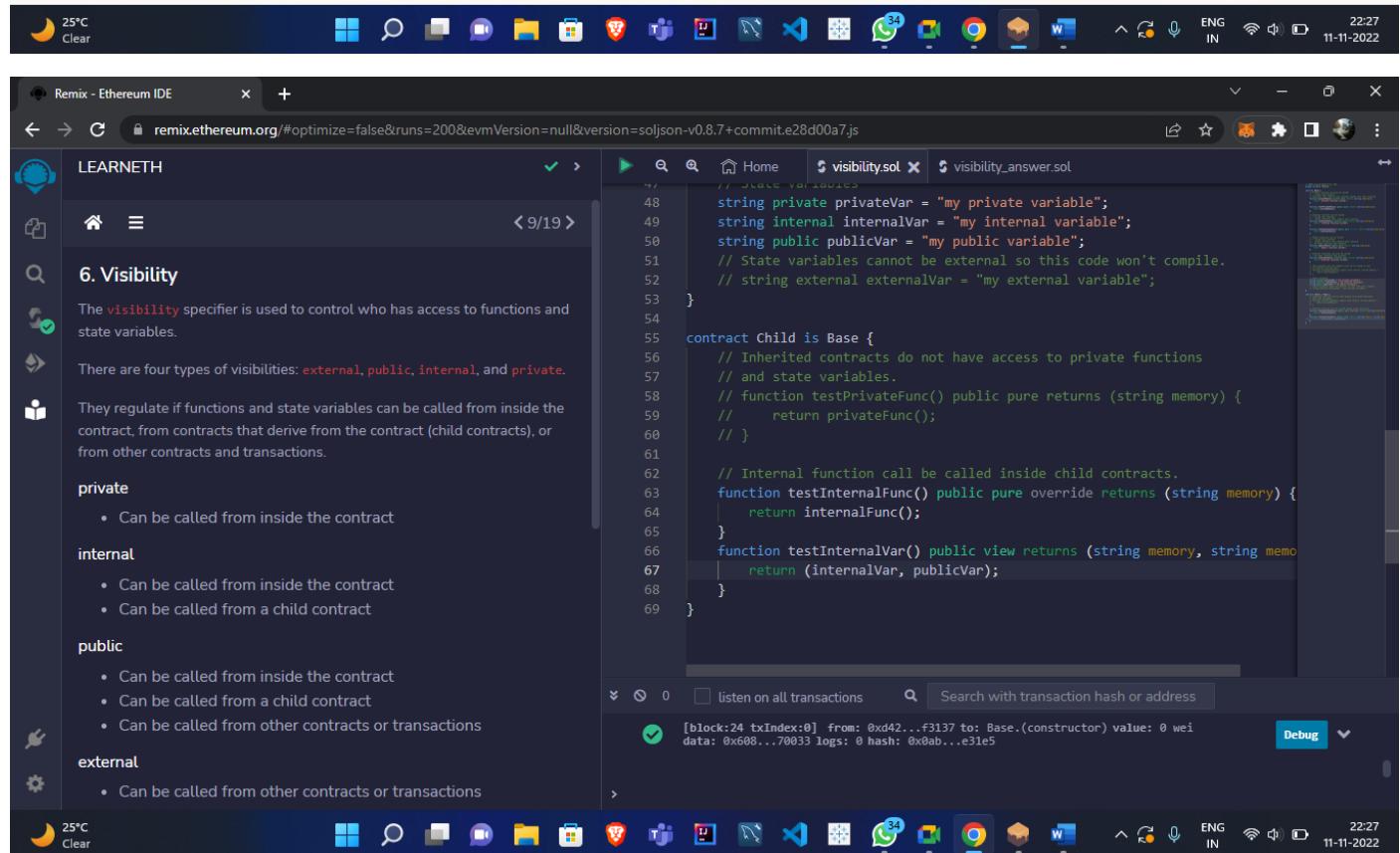
6. Visibility : Assignment

Create a new function in the Child contract called `testInternalVar` that returns the values of all state variables from the Base contract that are possible to return.



The screenshot shows the Ganache interface. At the top, there are tabs for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, and LOGS. Below the tabs, it displays the current block (24), gas price (20000000000), gas limit (6721975), hardfork (MUIRGLACIER), network ID (5777), RPC server (HTTP://127.0.0.1:8545), and mining status (AUTOMINING). A search bar at the top right allows searching for block numbers or transaction hashes. The main area shows a table of accounts with their addresses, balances, transaction counts, and indices. The mnemonic seed is also visible at the top.

ADDRESS	BALANCE	TX COUNT	INDEX	
0xC1E88a5CF1082DD0674F09eA3CE9CFda3b264d20	999.99 ETH	3	0	🔑
0xd42B0Ecec2BCe68f76b240f8C6696779859f3137	999.91 ETH	21	1	🔑
0xD306dFD52eea4bFbD23587EBDBe8EA71f68C1c2B	1000.00 ETH	0	2	🔑
0x13503F6B21c35Cd58Db63AF38B834807e4543Bc5	1000.00 ETH	0	3	🔑
0x49955354711B1Fc26B45a76004aA590e95F90dEe	1000.00 ETH	0	4	🔑
0xd4dA436579cdc60339ab41f22E69f44332655401	1000.00 ETH	0	5	🔑



The screenshot shows the Remix Ethereum IDE. The left sidebar has a navigation menu with 'LEARNETH' at the top, followed by '6. Visibility', 'The visibility specifier is used to control who has access to functions and state variables.', 'There are four types of visibilities: external, public, internal, and private.', 'They regulate if functions and state variables can be called from inside the contract, from contracts that derive from the contract (child contracts), or from other contracts and transactions.', and lists for 'private', 'internal', 'public', and 'external' visibility levels. The right side shows the Solidity code for 'visibility.sol' and 'visibility_answer.sol'. The code includes state variables and functions for testing them. A transaction log is visible at the bottom.

```
// State variables
string private privateVar = "my private variable";
string internal internalVar = "my internal variable";
string public publicVar = "my public variable";
// State variables cannot be external so this code won't compile.
// string external externalVar = "my external variable";

contract Child is Base {
    // Inherited contracts do not have access to private functions
    // and state variables.
    // function testPrivateFunc() public pure returns (string memory) {
    //     return privateFunc();
    // }

    // Internal function call be called inside child contracts.
    function testInternalFunc() public pure override returns (string memory) {
        return internalFunc();
    }

    function testInternalVar() public view returns (string memory, string memory) {
        return (internalVar, publicVar);
    }
}
```

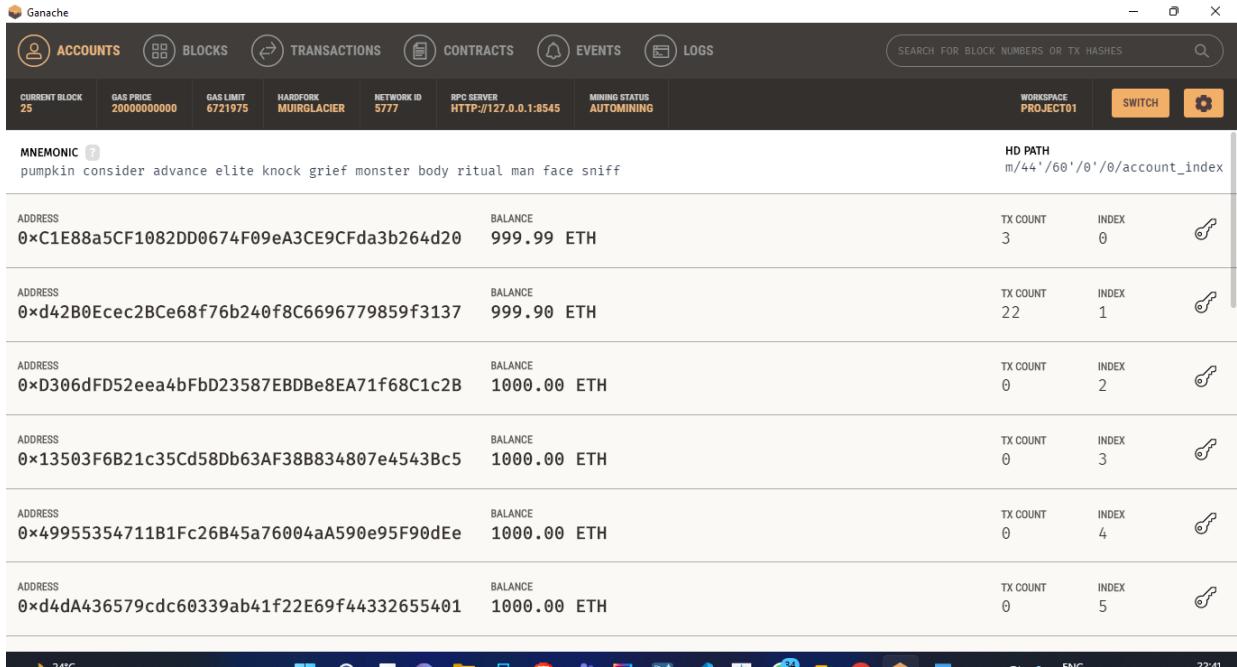
7.1 Control Flow - If/Else: Assignment

Create a new function called evenCheck in the IfElse contract:

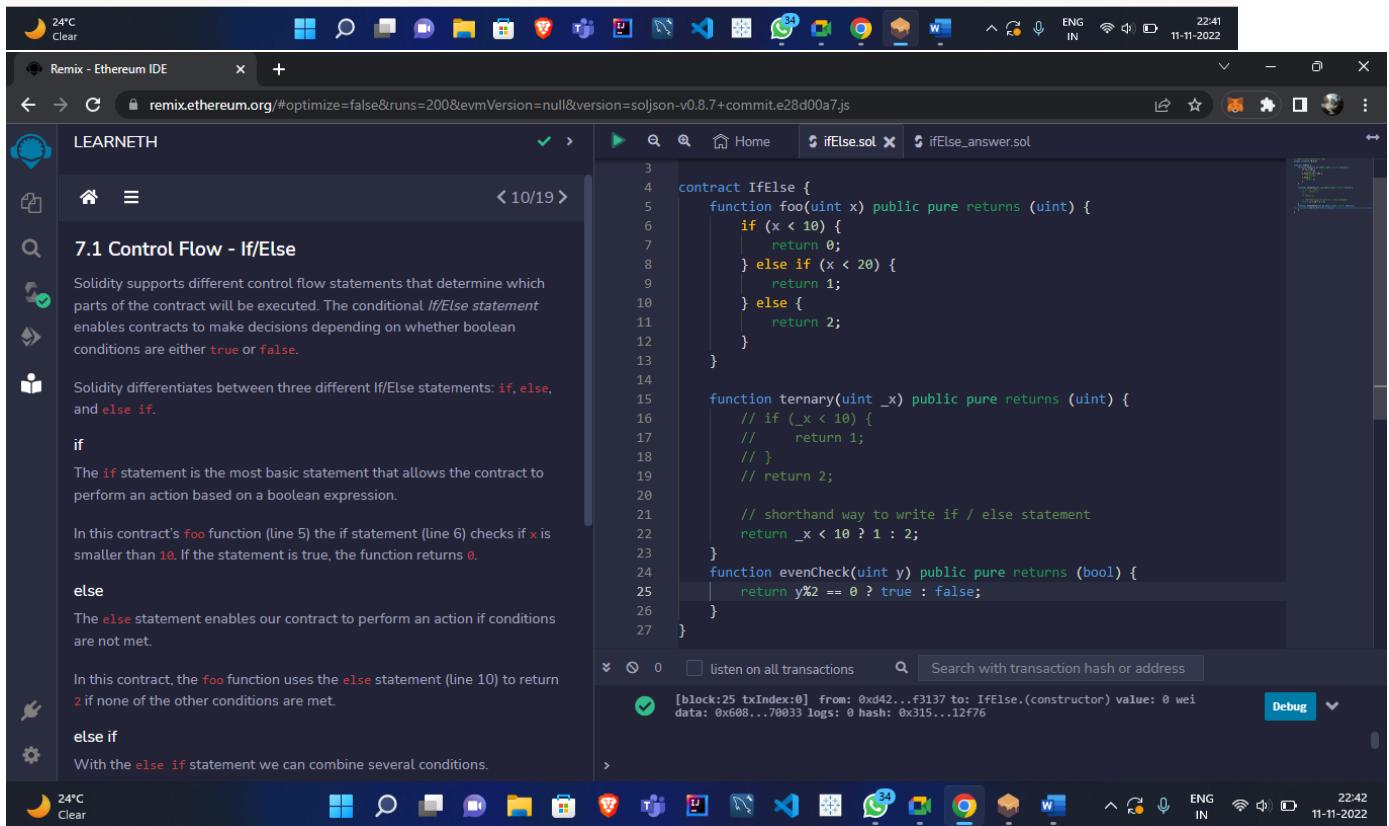
That takes in a uint as an argument.

The function returns true if the argument is even, and false if the argument is odd.

Use a ternary operator to return the result of the evenCheck function.



MNEMONIC	HD PATH
pumpkin consider advance elite knock grief monster body ritual man face sniff	m/44'/60'/0'/0/account_index
ADDRESS	
0xC1E88a5CF1082DD0674F09eA3CE9CFda3b264d20	BALANCE 999.99 ETH
ADDRESS	
0xd42B0Ecec2BCe68f76b240f8C6696779859f3137	BALANCE 999.90 ETH
ADDRESS	
0xD306dFD52eea4bFbD23587EBDBe8EA71f68C1c2B	BALANCE 1000.00 ETH
ADDRESS	
0x13503F6B21c35Cd58Db63AF38B834807e4543Bc5	BALANCE 1000.00 ETH
ADDRESS	
0x49955354711B1Fc26B45a76004aA590e95F90dEe	BALANCE 1000.00 ETH
ADDRESS	
0xd4dA436579cdc60339ab41f22E69f44332655401	BALANCE 1000.00 ETH



```
contract IfElse {
    function foo(uint x) public pure returns (uint) {
        if (x < 10) {
            return 0;
        } else if (x < 20) {
            return 1;
        } else {
            return 2;
        }
    }

    function ternary(uint _x) public pure returns (uint) {
        // if (_x < 10) {
        //     return 1;
        // }
        // return 2;

        // shorthand way to write if / else statement
        return _x < 10 ? 1 : 2;
    }

    function evenCheck(uint y) public pure returns (bool) {
        return y%2 == 0 ? true : false;
    }
}
```

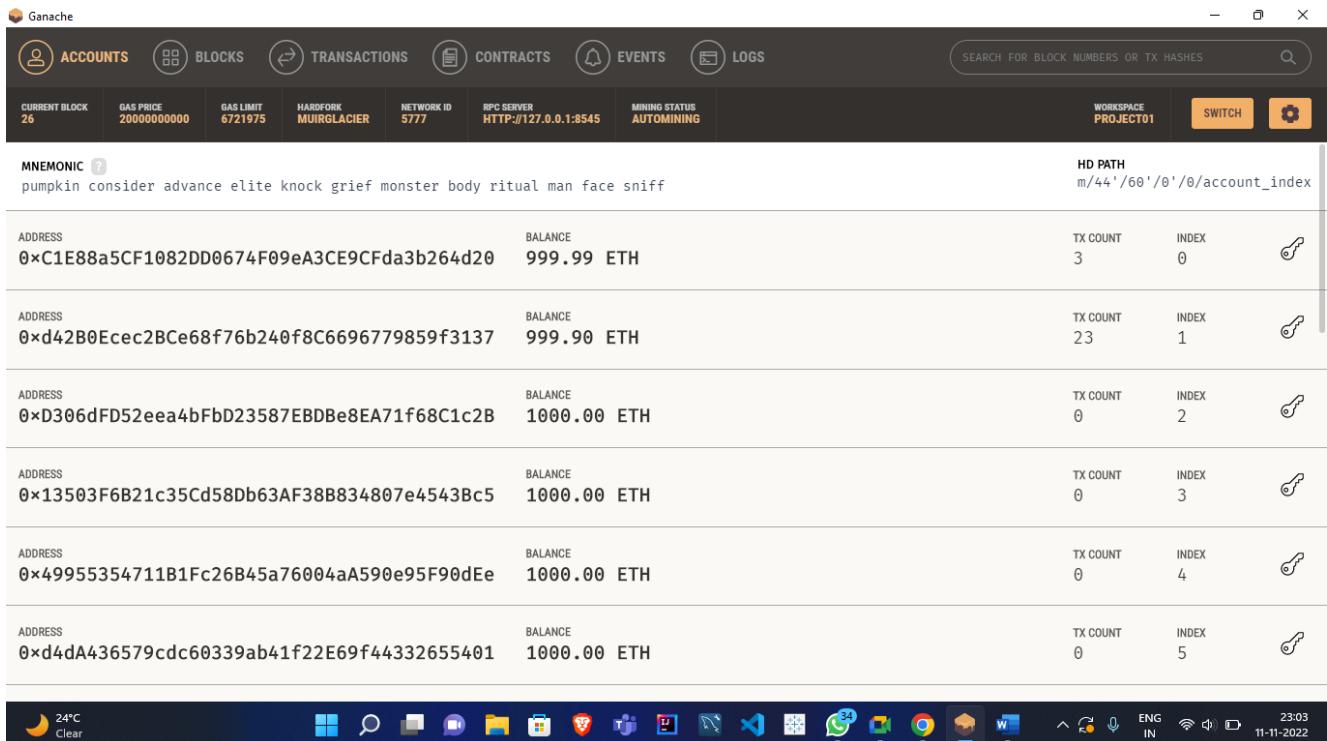
7.2 Control Flow – Loops: Assignment

Create a new function called evenCheck in the IfElse contract:

That takes in a uint as an argument.

The function returns true if the argument is even, and false if the argument is odd.

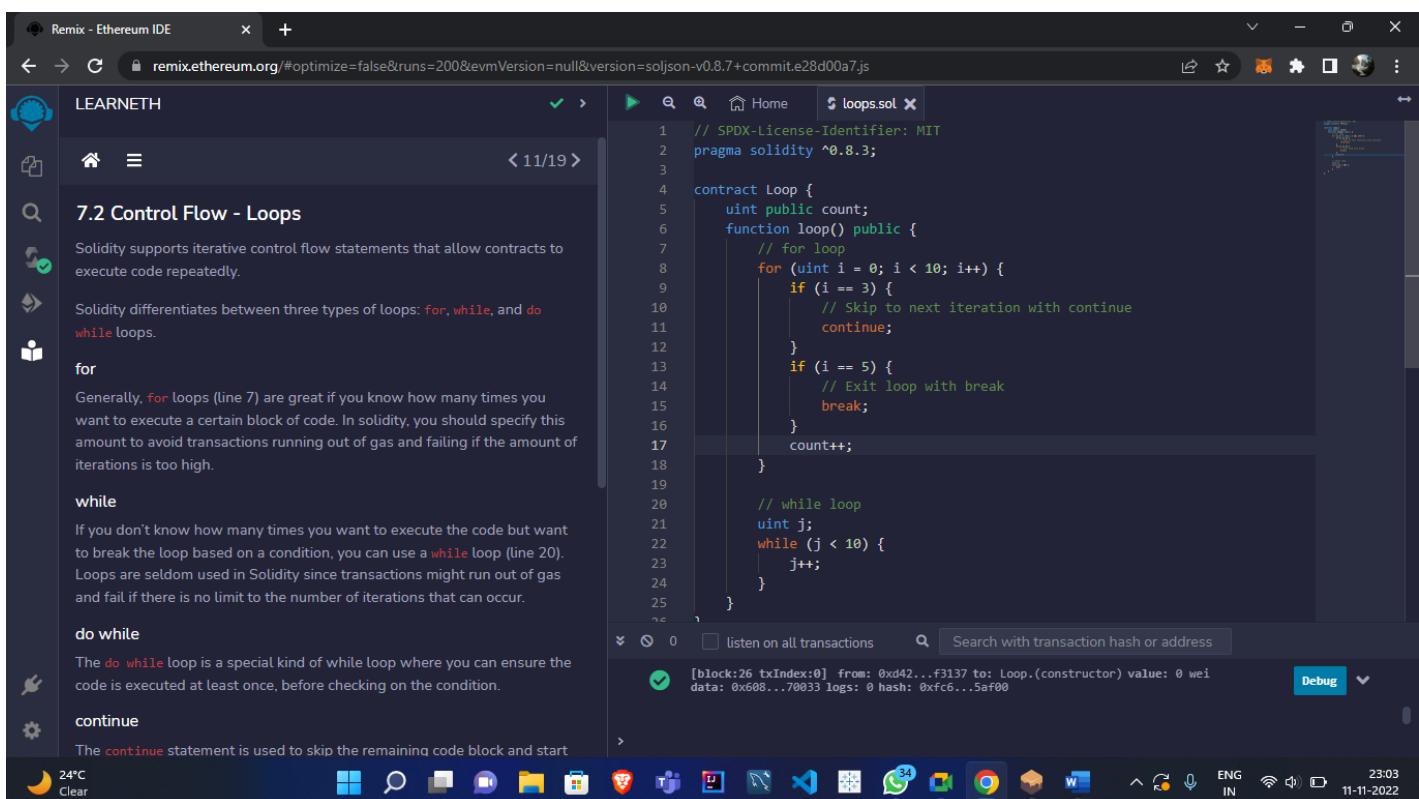
Use a ternary operator to return the result of the evenCheck function.



The screenshot shows the Ganache interface. At the top, there are tabs for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, and LOGS. Below these are network settings: CURRENT BLOCK (26), GAS PRICE (2000000000), GAS LIMIT (6721975), HARDFORK (MUIRGLACIER), NETWORK ID (5777), and RPC SERVER (HTTP://127.0.0.1:8545). The MINING STATUS is AUTOMINING. The workspace is set to PROJECT01. A search bar at the top right allows searching for block numbers or transaction hashes. The main area displays a mnemonic seed phrase: "pumpkin consider advance elite knock grief monster body ritual man face sniff". Below this, a table lists five accounts with their addresses, balances, transaction counts, and indices. Each account has a copy icon to its right. The accounts are:

ADDRESS	BALANCE	TX COUNT	INDEX
0xC1E88a5CF1082DD0674F09eA3CE9CFda3b264d20	999.99 ETH	3	0
0xd42B0Ecec2BCe68f76b240f8C6696779859f3137	999.90 ETH	23	1
0xD306dFD52eea4bFbD23587EBDBe8EA71f68C1c2B	1000.00 ETH	0	2
0x13503F6B21c35Cd58Db63AF38B834807e4543Bc5	1000.00 ETH	0	3
0x49955354711B1Fc26B45a76004aA590e95F90dEe	1000.00 ETH	0	4
0xd4dA436579cdc60339ab41f22E69f44332655401	1000.00 ETH	0	5

At the bottom, there is a Windows taskbar with various icons and a status bar showing the date and time (11-11-2022, 23:03).



The screenshot shows the Remix Ethereum IDE. The top bar shows the URL: remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js. The sidebar on the left is titled "7.2 Control Flow - Loops" and contains documentation for loops, including sections on "for", "while", and "do while" loops. The main editor area shows a Solidity contract named "Loop" with the following code:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;

contract Loop {
    uint public count;
    function loop() public {
        // for loop
        for (uint i = 0; i < 10; i++) {
            if (i == 3) {
                // Skip to next iteration with continue
                continue;
            }
            if (i == 5) {
                // Exit loop with break
                break;
            }
            count++;
        }
        // while loop
        uint j;
        while (j < 10) {
            j++;
        }
    }
}
```

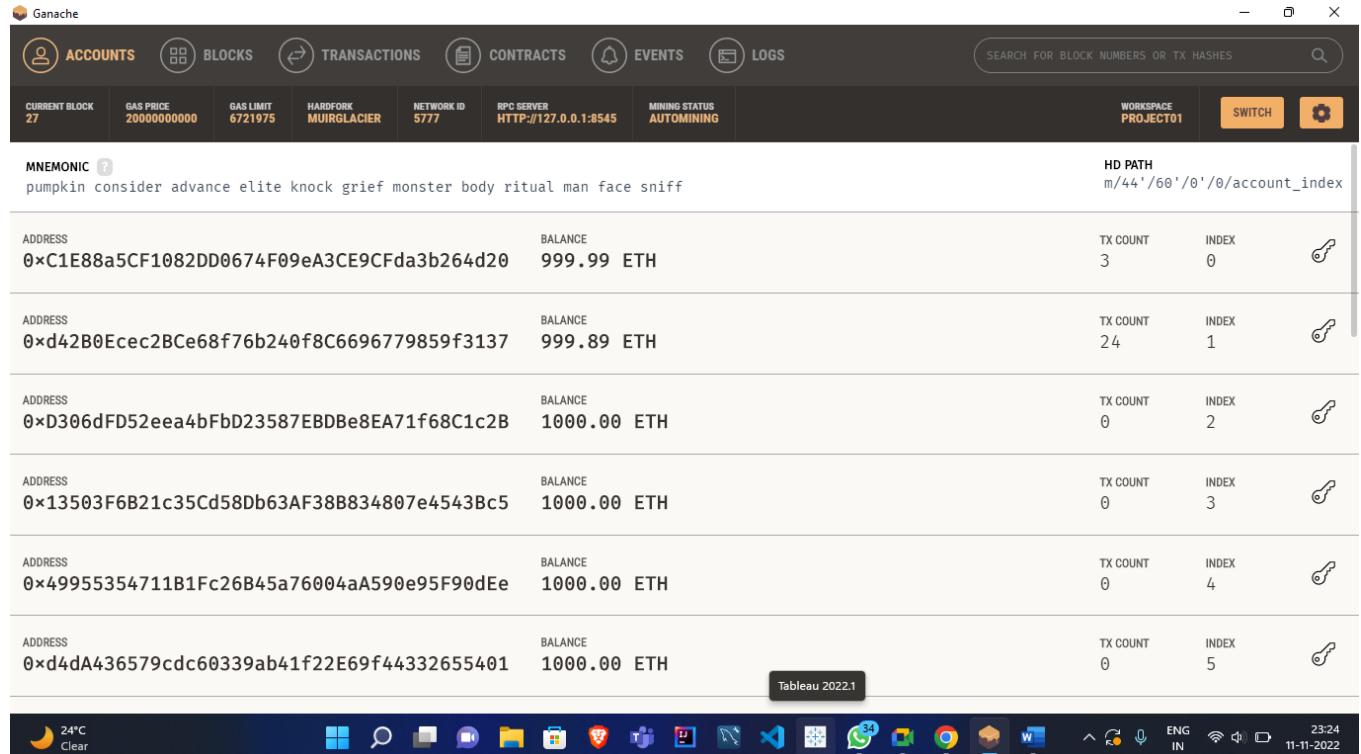
The status bar at the bottom shows the date and time (11-11-2022, 23:03) and a transaction log: [block:26 txIndex:0] from: 0xd42...f3137 to: Loop.(constructor) value: 0 wei data: 0x608...70033 logs: 0 hash: 0xfc6...5af00. A "Debug" button is also visible in the bottom right.

8.1 Data Structures – Arrays: Assignment

Initialize a public fixed-sized array called arr3 with the values 0, 1,

2. Make the size as small as possible.

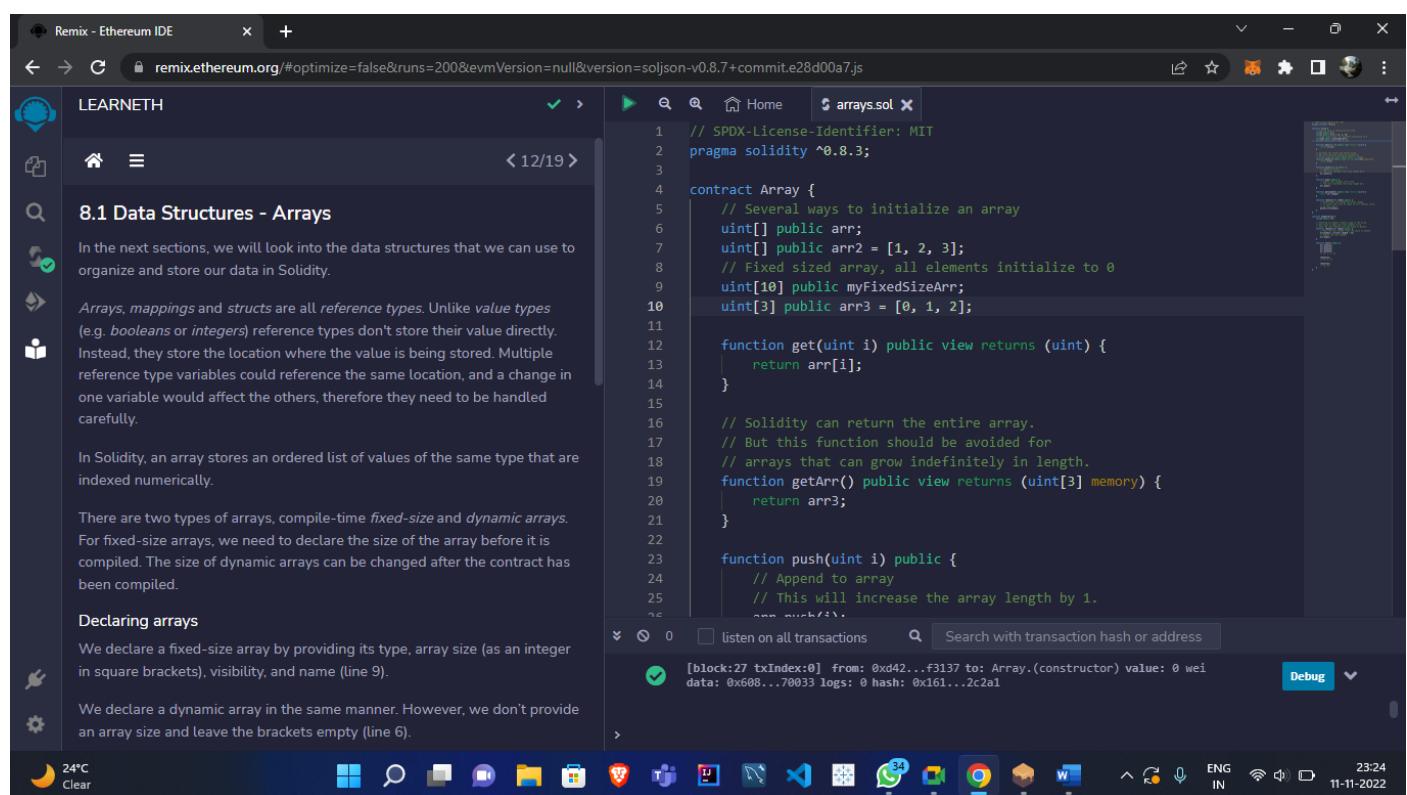
Change the getArr() function to return the value of arr3.



The screenshot shows the Ganache interface. At the top, there are tabs for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, and LOGS. Below these are various status indicators: CURRENT BLOCK (27), GAS PRICE (20000000000), GAS LIMIT (6721975), HARDFORK (MUIRGLACIER), NETWORK ID (5777), and RPC SERVER (HTTP://127.0.0.1:8545). The MINING STATUS is AUTOMINING. On the right, there is a WORKSPACE PROJECT01 and a SWITCH button. Below the header, the MNEMONIC is listed as "pumpkin consider advance elite knock grief monster body ritual man face sniff". The HD PATH is m/44'/60'/0'/0/account_index. The main table lists five accounts with their addresses, balances, transaction counts, and indices. The accounts are:

ADDRESS	BALANCE	TX COUNT	INDEX
0xC1E88a5CF1082DD0674F09eA3CE9CFda3b264d20	999.99 ETH	3	0
0xd42B0Ecec2BCe68f76b240f8C6696779859f3137	999.89 ETH	24	1
0xD306dFD52eea4bFbD23587EBDBe8EA71f68C1c2B	1000.00 ETH	0	2
0x13503F6B21c35Cd58Db63AF38B834807e4543Bc5	1000.00 ETH	0	3
0x49955354711B1Fc26B45a76004aA590e95F90dEe	1000.00 ETH	0	4
0xd4dA436579cdc60339ab41f22E69f44332655401	1000.00 ETH	0	5

At the bottom of the table, a "Tableau 2022.1" watermark is visible. The system tray at the bottom shows the date and time as 23:24 11-11-2022.



The screenshot shows the Remix - Ethereum IDE interface. The left sidebar has a "LEARNETH" section with a "8.1 Data Structures - Arrays" article. The article discusses arrays, mappings, and structs as reference types, noting they store the location of the value. It also explains fixed-size and dynamic arrays, and how to declare arrays. The right side shows the Solidity code for an "Array" contract:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;

contract Array {
    // Several ways to initialize an array
    uint[] public arr;
    uint[] public arr2 = [1, 2, 3];
    // Fixed sized array, all elements initialize to 0
    uint[10] public myFixedSizeArr;
    uint[3] public arr3 = [0, 1, 2];

    function get(uint i) public view returns (uint) {
        return arr[i];
    }

    // Solidity can return the entire array.
    // But this function should be avoided for
    // arrays that can grow indefinitely in length.
    function getArr() public view returns (uint[3] memory) {
        return arr3;
    }

    function push(uint i) public {
        // Append to array
        // This will increase the array length by 1.
        arr.push(i);
    }
}
```

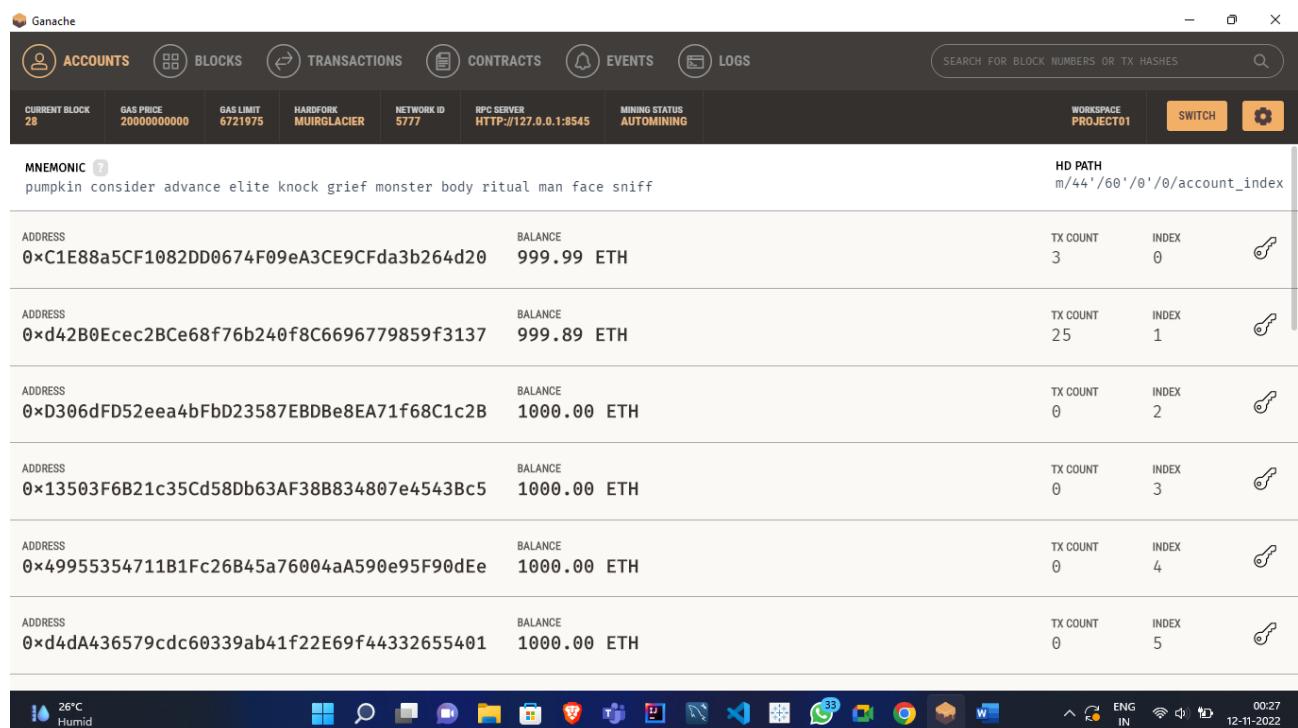
The status bar at the bottom shows the date and time as 23:24 11-11-2022.

8.2 Data Structures – Mappings: Assignment

Create a public mapping balances that associates the key type address with the value type uint.

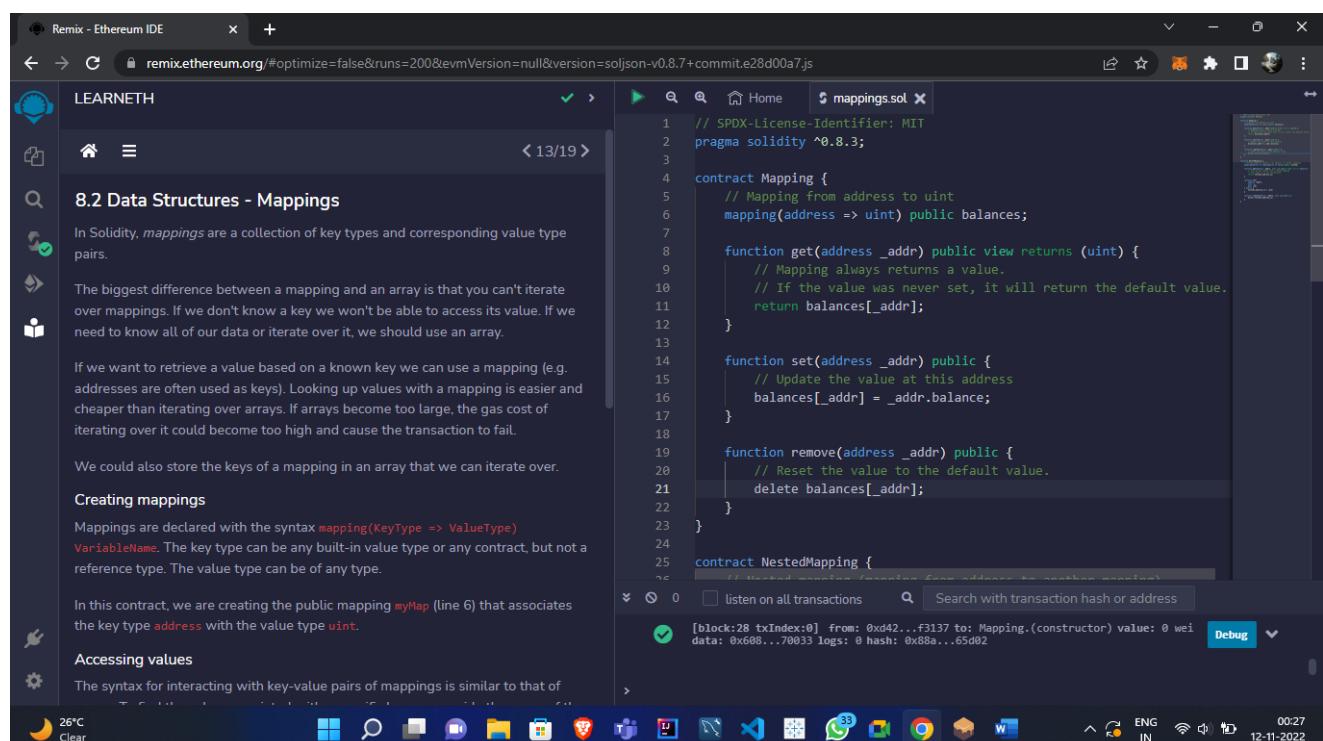
Change the functions get and remove to work with the mapping balances.

Change the function set to create a new entry to the balances mapping, where the key is the address of the parameter and the value is the balance associated with the address of the parameter.



The screenshot shows the Ganache interface with the following details:

ADDRESS	BALANCE	TX COUNT	INDEX	EDIT
0xC1E88a5CF1082DD0674F09eA3CE9CFda3b264d20	999.99 ETH	3	0	🔗
0xd42B0Ecec2BCe68f76b240f8C6696779859f3137	999.89 ETH	25	1	🔗
0xD306dFD52eea4bFbD23587EBDBe8EA71f68C1c2B	1000.00 ETH	0	2	🔗
0x13503F6B21c35Cd58Db63AF38B834807e4543Bc5	1000.00 ETH	0	3	🔗
0x49955354711B1Fc26B45a76004aA590e95F90dEe	1000.00 ETH	0	4	🔗
0xd4dA436579cdc60339ab41f22E69f44332655401	1000.00 ETH	0	5	🔗



The screenshot shows the Remix IDE interface with the following details:

- Left Sidebar:** Shows a navigation tree with sections like LEARNETH, 8.2 Data Structures - Mappings, Creating mappings, Accessing values, and a note about mapping to arrays.
- Right Sidebar:** Shows the current weather (26°C, Humid) and a system tray with various icons.
- Code Editor:** Displays the Solidity code for a mapping contract:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;

contract Mapping {
    // Mapping from address to uint
    mapping(address => uint) public balances;

    function get(address _addr) public view returns (uint) {
        // Mapping always returns a value.
        // If the value was never set, it will return the default value.
        return balances[_addr];
    }

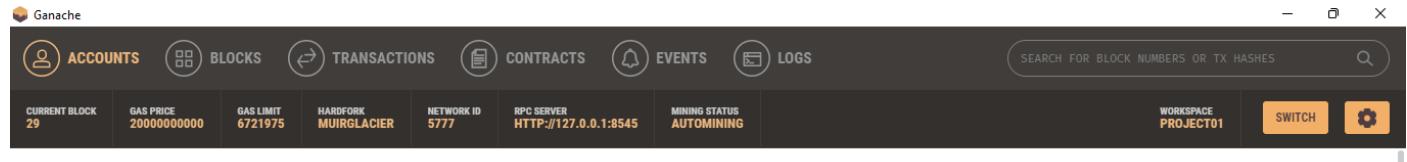
    function set(address _addr) public {
        // Update the value at this address
        balances[_addr] = _addr.balance;
    }

    function remove(address _addr) public {
        // Reset the value to the default value.
        delete balances[_addr];
    }
}

contract NestedMapping {
```
- Bottom Status Bar:** Shows the current block number (28), transaction index (0), and a log entry: [block:28 txIndex:0] from: 0xd42...f3137 to: Mapping.(constructor) value: 0 wei data: 0x608...70033 logs: 0 hash: 0x88a...65d02.

8.3 Data Structures – Structs: Assignment

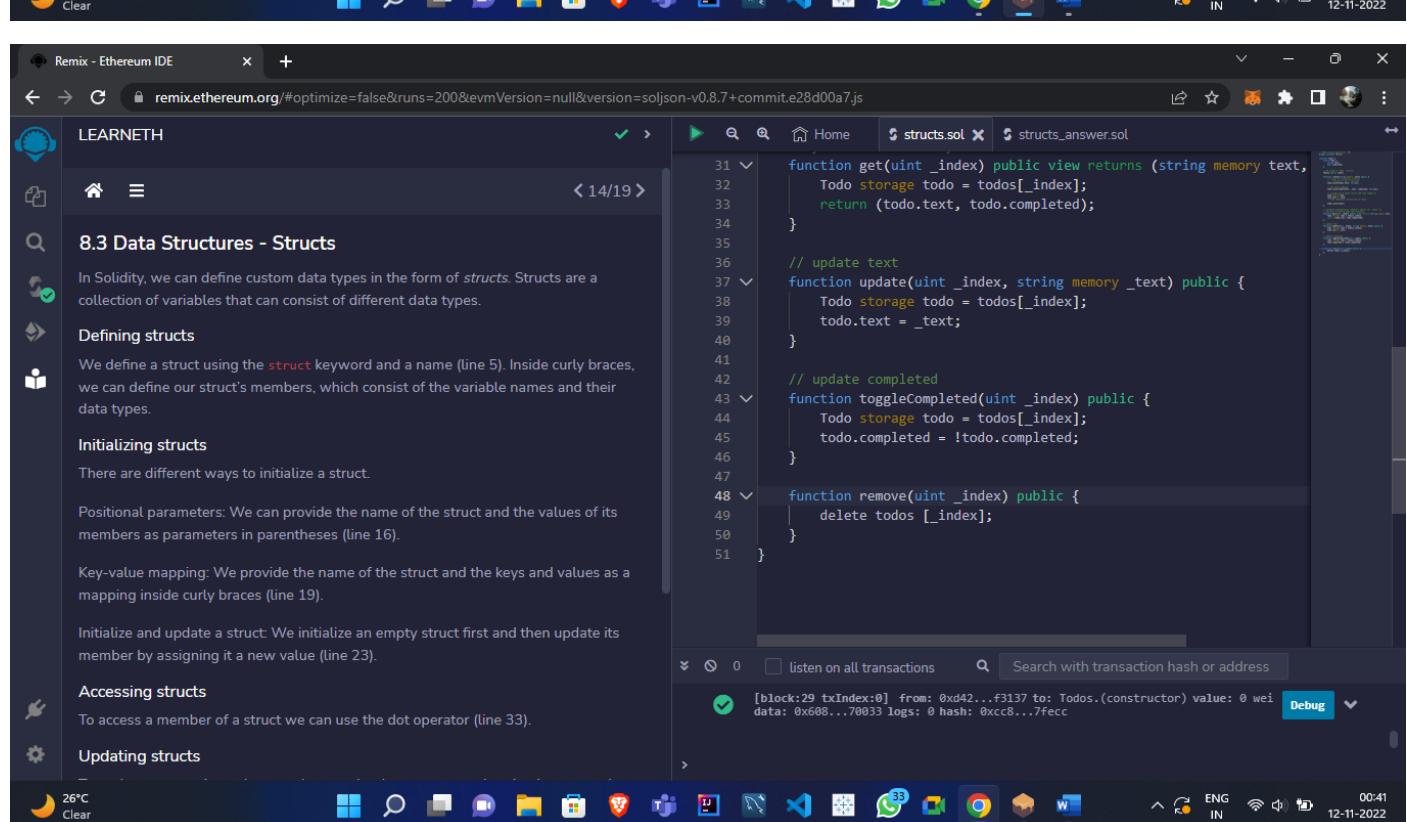
Create a function `remove` that takes a `uint` as a parameter and deletes a struct member with the given index in the todos mapping.



MNEMONIC: pumpkin consider advance elite knock grief monster body ritual man face sniff

HD PATH: m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0xC1E88a5CF1082DD0674F09eA3CE9CFda3b264d20	999.99 ETH	3	0
0xd42B0Ecec2BCe68f76b240f8C6696779859f3137	999.87 ETH	26	1
0xD306dFD52eea4bFbD23587EBDBe8EA71f68C1c2B	1000.00 ETH	0	2
0x13503F6B21c35Cd58Db63AF38B834807e4543Bc5	1000.00 ETH	0	3
0x49955354711B1Fc26B45a76004aA590e95F90dEe	1000.00 ETH	0	4
0xd4dA436579cdc60339ab41f22E69f44332655401	1000.00 ETH	0	5



```
function get(uint _index) public view returns (string memory text, Todo storage todo) {
    todo = todos[_index];
    return (todo.text, todo.completed);
}

// update text
function update(uint _index, string memory _text) public {
    todo = todos[_index];
    todo.text = _text;
}

// update completed
function toggleCompleted(uint _index) public {
    todo = todos[_index];
    todo.completed = !todo.completed;
}

function remove(uint _index) public {
    delete todos[_index];
}
```

Remix - Ethereum IDE

26°C Clear

8.3 Data Structures - Structs

In Solidity, we can define custom data types in the form of `structs`. Structs are a collection of variables that can consist of different data types.

Defining structs

We define a struct using the `struct` keyword and a name (line 5). Inside curly braces, we can define our struct's members, which consist of the variable names and their data types.

Initializing structs

There are different ways to initialize a struct.

Positional parameters: We can provide the name of the struct and the values of its members as parameters in parentheses (line 16).

Key-value mapping: We provide the name of the struct and the keys and values as a mapping inside curly braces (line 19).

Initialize and update a struct: We initialize an empty struct first and then update its member by assigning it a new value (line 23).

Accessing structs

To access a member of a struct we can use the dot operator (line 33).

Updating structs

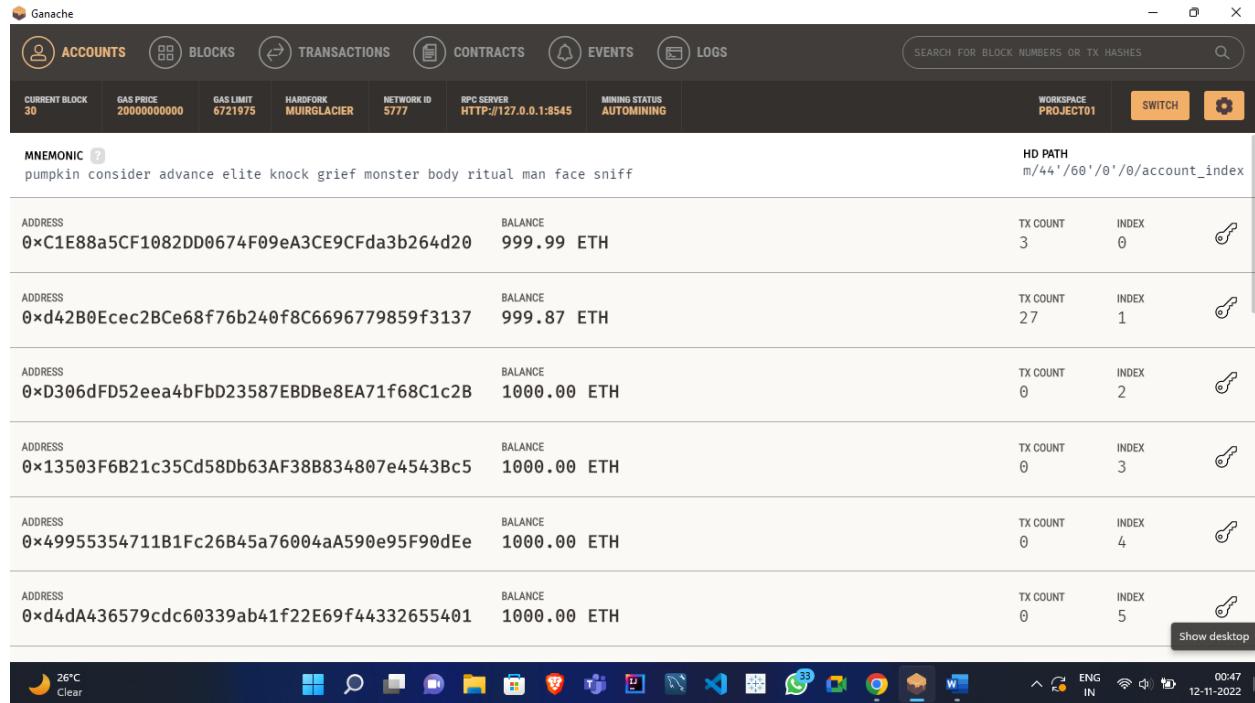
26°C Clear

8.4 Data Structures – Enums: Assignment

Define an enum type called **Size** with the members **S**, **M**, and **L**.

Initialize the variable sizes of the enum type **Size**.

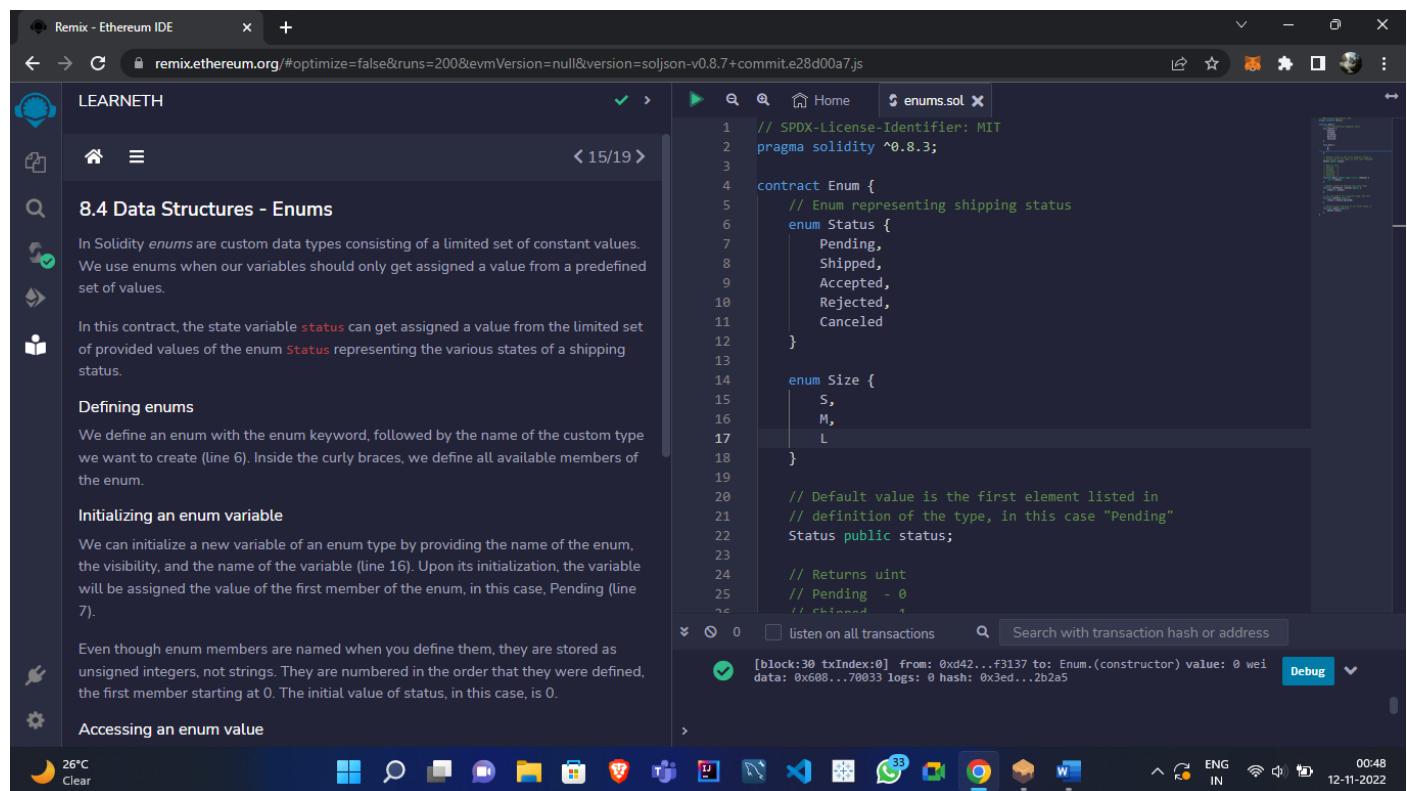
Create a getter function **getSize()** that returns the value of the variable sizes.



The screenshot shows the Ganache interface. At the top, there are tabs for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, and LOGS. Below these are status indicators for CURRENT BLOCK (30), GAS PRICE (2000000000), GAS LIMIT (6721975), HARDFORK (MUIRGLACIER), NETWORK ID (5777), and RPC SERVER (HTTP://127.0.0.1:8545). The MINING STATUS is AUTOMINING. The WORKSPACE is set to PROJECT01. A search bar at the top right allows searching for block numbers or tx hashes. The main area displays a mnemonic seed phrase: "pumpkin consider advance elite knock grief monster body ritual man face sniff". Below this, a table lists five accounts with their addresses, balances, transaction counts, and indices. The accounts are:

ADDRESS	BALANCE	TX COUNT	INDEX
0xC1E88a5CF1082DD0674F09eA3CE9CFda3b264d20	999.99 ETH	3	0
0xd42B0Ecec2BCe68f76b240f8C6696779859f3137	999.87 ETH	27	1
0xD306dFD52eea4bFbD23587EBDBe8EA71f68C1c2B	1000.00 ETH	0	2
0x13503F6B21c35Cd58Db63AF38B834807e4543Bc5	1000.00 ETH	0	3
0x49955354711B1Fc26B45a76004aA590e95F90dEe	1000.00 ETH	0	4
0xd4dA436579cdc60339ab41f22E69f44332655401	1000.00 ETH	0	5

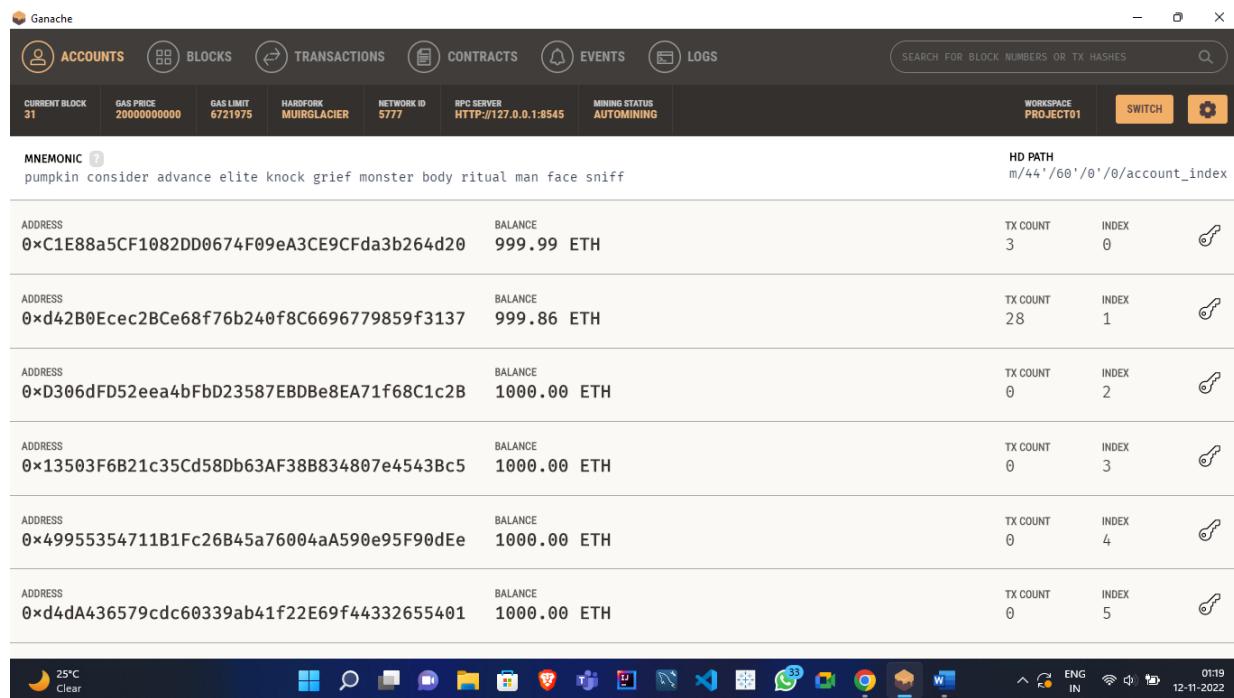
At the bottom, there is a "Show desktop" button and a weather widget showing 26°C and Clear. The taskbar at the bottom of the screen also shows various application icons.



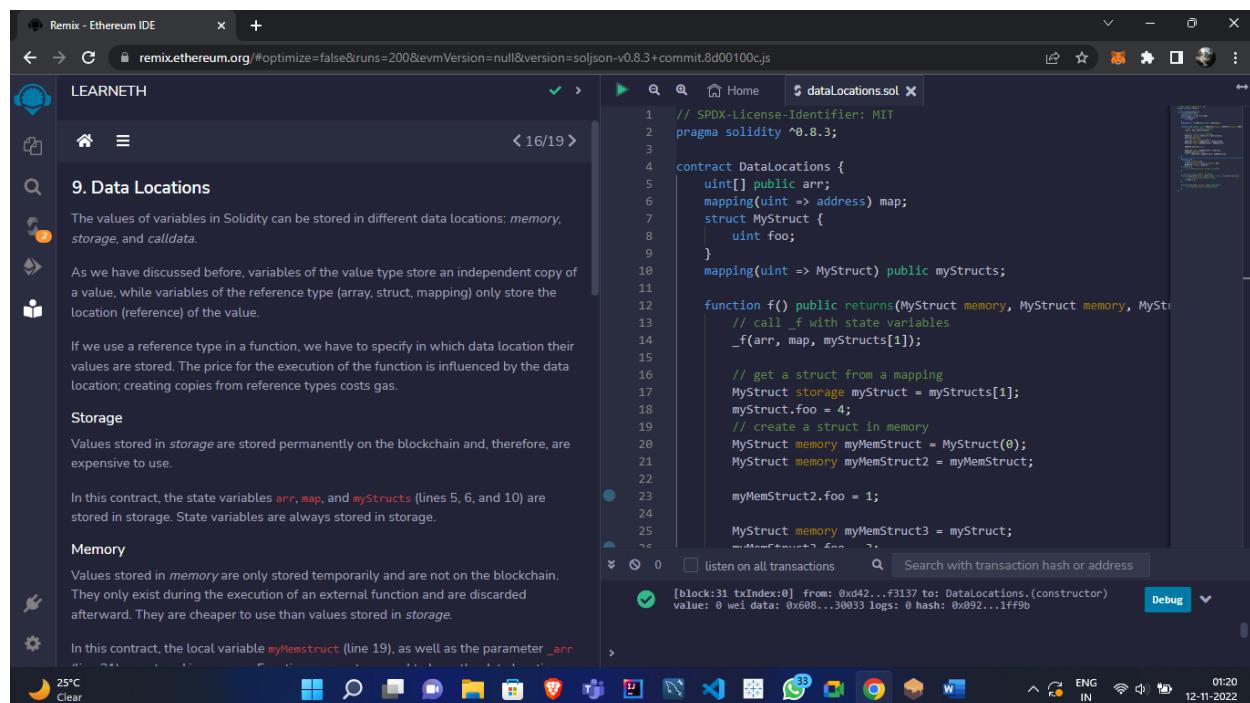
The screenshot shows the Remix - Ethereum IDE interface. The left sidebar has a "LEARNEETH" profile and a "8.4 Data Structures - Enums" section. The main area shows a Solidity code editor with the file "enums.sol". The code defines an enum **Status** with members **Pending**, **Shipped**, **Accepted**, **Rejected**, and **Canceled**. It also defines an enum **Size** with members **S**, **M**, and **L**. A note indicates that the default value is the first element listed in the definition. The code also includes a **status** variable of type **Status** and a **getSize** function that returns the value of the **status** variable. The bottom of the screen shows a terminal window with a transaction log and a "Debug" button, along with the Windows taskbar.

9. Data Locations: Assignment

1) Change the value of the `myStruct` member `foo`, inside the function `f`, to 4. 2) Create a new struct `myMemStruct2` with the data location `memory` inside the function `f` and assign it the value of `myMemStruct`. 3) Change the value of the `myMemStruct2` member `foo` to 1. Create a new struct `myMemStruct3` with the data location `memory` inside the function `f` and assign it the value of `myStruct`. Change the value of the `myMemStruct3` member `foo` to 3. 4) Let the function `f` return `myStruct`, `myMemStruct2`, and `myMemStruct3`.



The screenshot shows the Ganache interface. At the top, there are tabs for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, and LOGS. Below these are buttons for CURRENT BLOCK (31), GAS PRICE (20000000000), GAS LIMIT (6721975), HARDFORK (MUIRGLEACIER), NETWORK ID (5777), RPC SERVER (HTTP://127.0.0.1:8545), and MINING STATUS (AUTOMINING). A workspace switcher shows PROJECT01. The main area displays account details with a mnemonic seed phrase: "pumpkin consider advance elite knock grief monster body ritual man face sniff". The HD PATH is listed as m/44'/60'/0'/0/account_index. A table lists six accounts with their addresses, balances (999.99, 999.86, 1000.00, 1000.00, 1000.00, 1000.00 ETH), transaction counts (3, 28, 0, 0, 0, 0), and indices (0, 1, 2, 3, 4, 5). Each account has a copy icon. The bottom of the interface shows a taskbar with various icons and system status.



The screenshot shows the Remix - Ethereum IDE interface. The top bar shows the URL: remixethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.3+commit.8d00100cjs. The sidebar on the left has sections for LEARNETH, Data Locations, Storage, and Memory. The main code editor shows the `dataLocations.sol` contract:`// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;

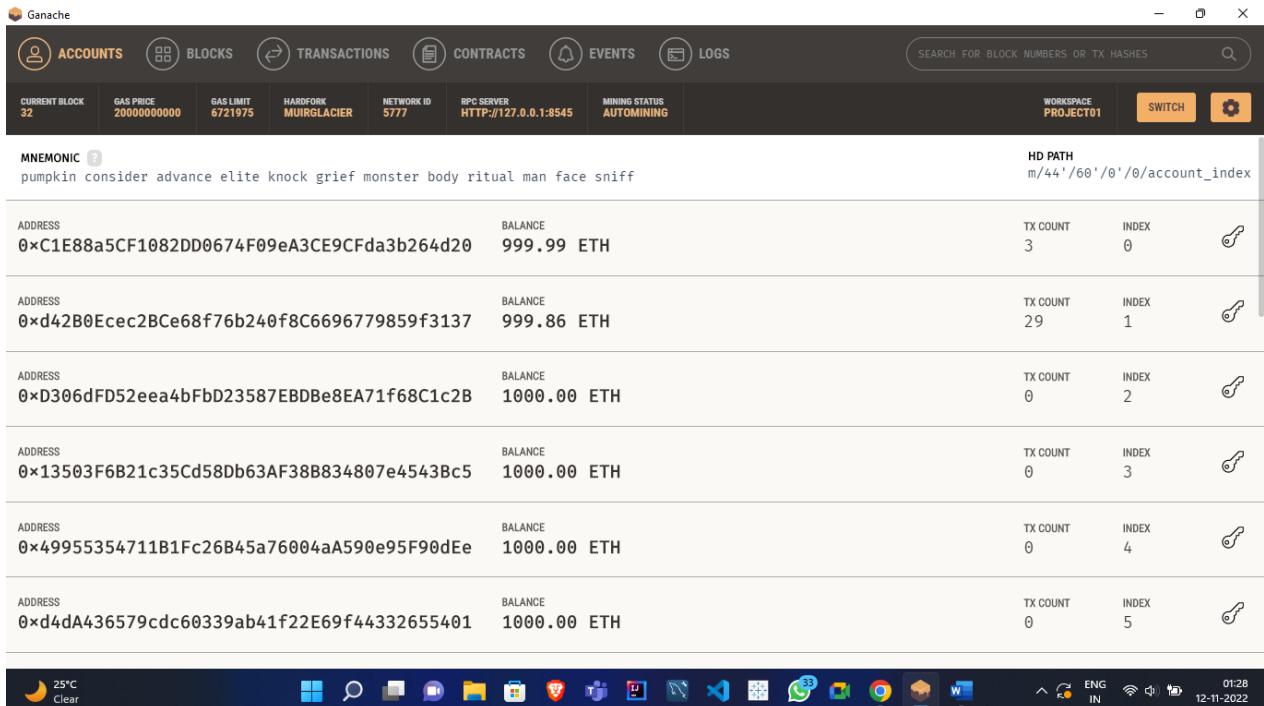
contract DataLocations {
 uint[] public arr;
 mapping(uint => address) map;
 struct MyStruct {
 uint foo;
 }
 mapping(uint => MyStruct) public myStructs;

 function f() public returns(MyStruct memory, MyStruct memory, MyStruct memory) {
 // call _f with state variables
 _f(arr, map, myStructs[1]);
 // get a struct from a mapping
 MyStruct storage myStruct = myStructs[1];
 myStruct.foo = 4;
 // create a struct in memory
 MyStruct memory myMemStruct = MyStruct(0);
 MyStruct memory myMemStruct2 = myMemStruct;
 myMemStruct2.foo = 1;
 MyStruct memory myMemStruct3 = myStruct;
 myMemStruct3.foo = 3;
 }
}` The bottom status bar shows the date and time as 12-11-2022 01:19.

10.1 Transactions - Ether and Wei:

Create a public uint called `oneGWei` and set it to 1 gwei.

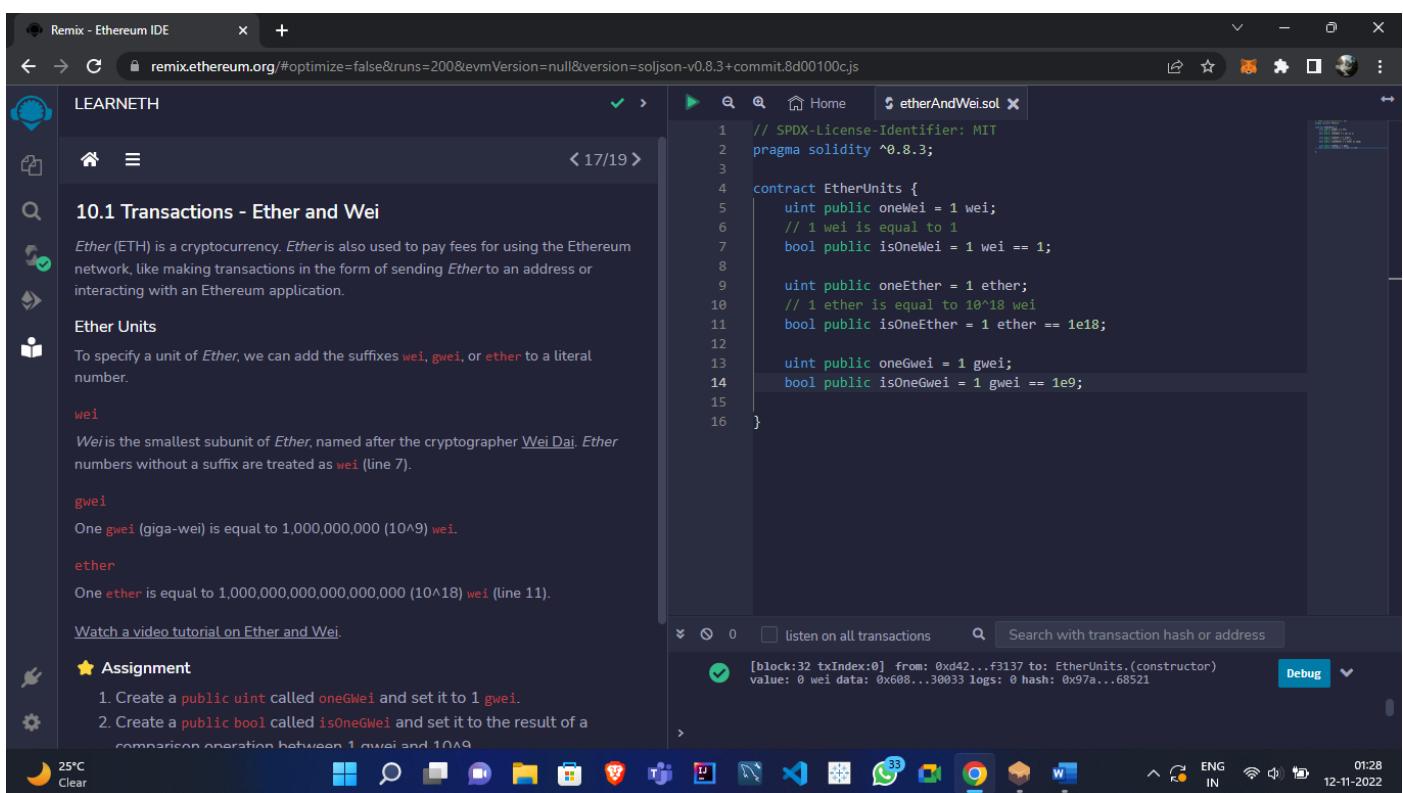
Create a public bool called `isOneGWei` and set it to the result of a comparison operation between 1 gwei and 10^9 .



MNEMONIC: pumpkin consider advance elite knock grief monster body ritual man face sniff

HD PATH: m/44'/60'/0/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0xC1E88a5CF1082DD0674F09eA3CE9CFda3b264d20	999.99 ETH	3	0
0xd42B0Ecec2BCe68f76b240f8C6696779859f3137	999.86 ETH	29	1
0xD306dFD52eea4bFbd23587EBDBe8EA71f68C1c2B	1000.00 ETH	0	2
0x13503F6B21c35Cd58Db63AF38B834807e4543Bc5	1000.00 ETH	0	3
0x49955354711B1Fc26B45a76004aA590e95F90dEe	1000.00 ETH	0	4
0xd4dA436579cdc60339ab41f22E69f44332655401	1000.00 ETH	0	5



```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.3;
3
4 contract EtherUnits {
5     uint public oneWei = 1 wei;
6     // 1 wei is equal to 1
7     bool public isOneWei = 1 wei == 1;
8
9     uint public oneEther = 1 ether;
10    // 1 ether is equal to 10^18 wei
11    bool public isOneEther = 1 ether == 1e18;
12
13    uint public oneGWei = 1 gwei;
14    bool public isOneGWei = 1 gwei == 1e9;
15
16 }
```

10.2 Transactions - Gas and Gas Price: Assignment

Create a new public state variable in the Gas contract called `cost` of the type `uint`. Store the value of the gas cost for deploying the contract in the new variable, including the cost for the value you are storing.

Ganache

MNEMONIC: pumpkin consider advance elite knock grief monster body ritual man face sniff

HD PATH: m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX	
0xC1E88a5CF1082DD0674F09eA3CE9CFda3b264d20	999.99 ETH	3	0	🔗
0xd42B0Ecec2BCe68f76b240f8C6696779859f3137	999.85 ETH	30	1	🔗
0xD306dFD52eea4bFbD23587EBDBe8EA71f68C1c2B	1000.00 ETH	0	2	🔗
0x13503F6B21c35Cd58Db63AF38B834807e4543Bc5	1000.00 ETH	0	3	🔗
0x49955354711B1Fc26B45a76004aA590e95F90dEe	1000.00 ETH	0	4	🔗
0xd4dA436579cdc60339ab41f22E69f44332655401	1000.00 ETH	0	5	🔗



Remix - Ethereum IDE

LEARNETH

When sending a transaction, the sender has to pay the `gas fee` (`gas_price * gas`) upon execution of the transaction. If `gas` is left over after the execution is completed, the sender gets refunded.

Gas prices are denoted in gwei.

Gas limit

When sending a transaction, the sender specifies the maximum amount of gas that they are willing to pay for. If they set the limit too low, their transaction can run out of `gas` before being completed, reverting any changes being made. In this case, the `gas` was consumed and can't be refunded.

Learn more about `gas` on ethereum.org.

Watch a video tutorial on [Gas and Gas Price](#).

★ Assignment

Create a new `public` state variable in the `Gas` contract called `cost` of the type `uint`. Store the value of the gas cost for deploying the contract in the new variable, including the cost for the value you are storing.

Tip: You can check in the Remix terminal the details of a transaction, including the gas cost. You can also use the Remix plugin `Gas Profiler` to check for the gas cost of transactions.

Check Answer Show answer

GasAndGasPrice.sol

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.3;
3
4 contract Gas {
5     uint public i = 0;
6     uint public cost = 170367;
7     // Using up all of the gas that you send causes your transaction to fail.
8     // State changes are undone.
9     // Gas spent are not refunded.
10    function forever() public {
11        // Here we run a loop until all of the gas are spent
12        // and the transaction fails
13        while (true) {
14            i += 1;
15        }
16    }
17 }
```

0 listen on all transactions Search with transaction hash or address

[block:33 txIndex:0] from: 0xd42...f3137 to: Gas.(constructor) value: 0 wei data: 0x608...30033 logs: 0 hash: 0x478...8e0b Debug

25°C Clear

Windows taskbar showing the Remix IDE icon and system status icons.

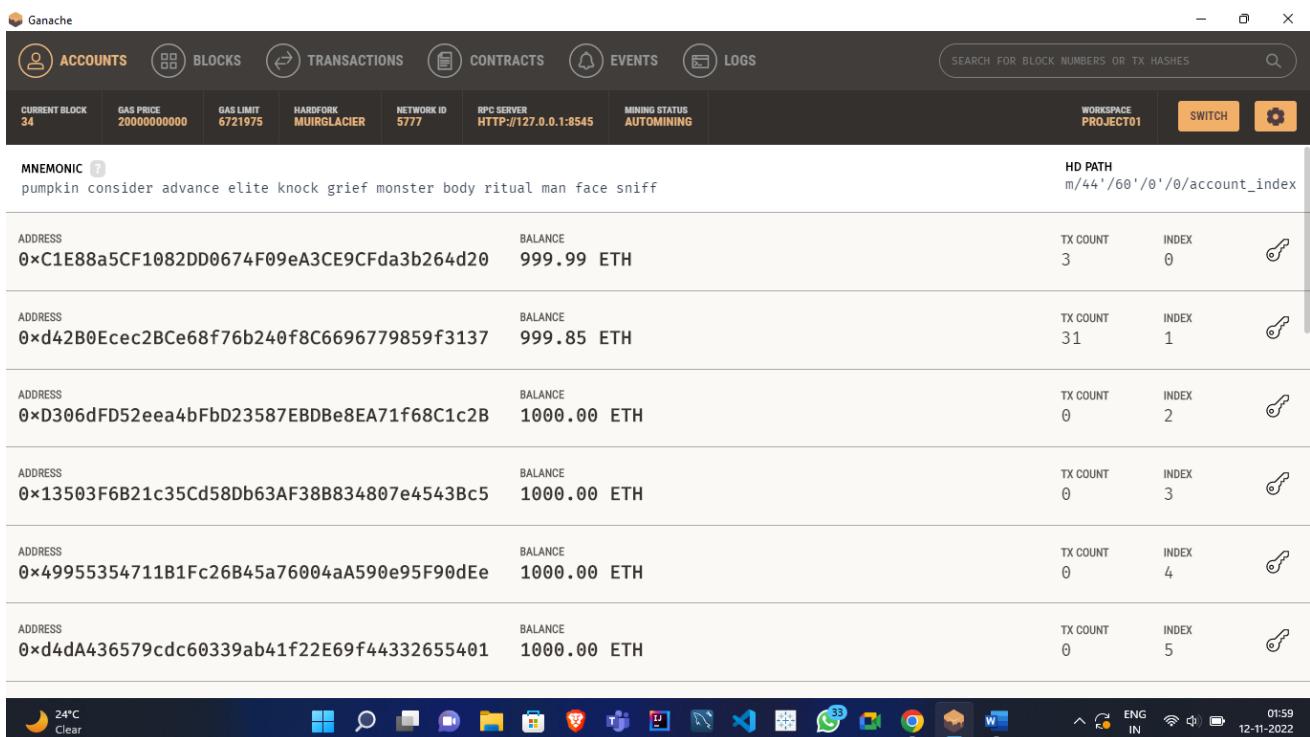
10.3 Transactions - Sending Ether: Assignment

Build a charity contract that receives Ether that can be withdrawn by a beneficiary. Create a contract called Charity.

Add a public state variable called owner of the type address.

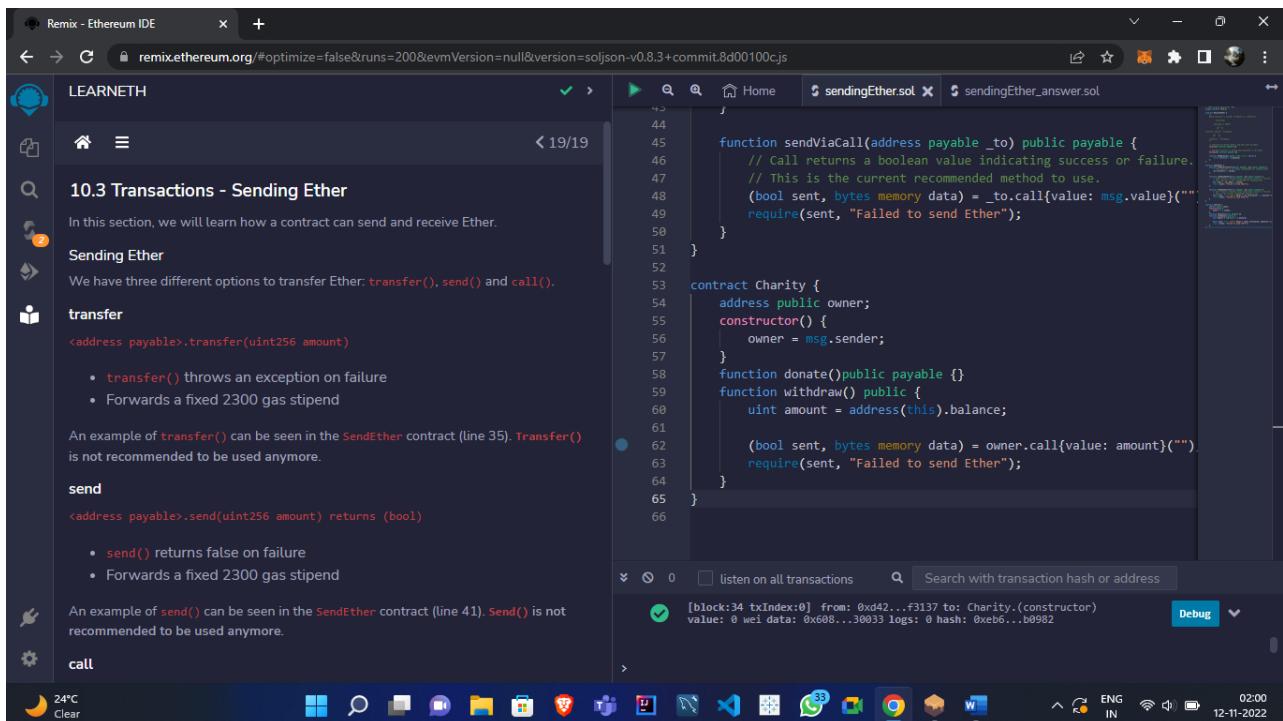
Create a donate function that is public and payable without any parameters or function code.

Create a withdraw function that is public and sends the total balance of the contract to the owner address.



The screenshot shows the Ganache interface. At the top, there are tabs for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, and LOGS. Below the tabs, it displays the current block (34), gas price (2000000000), gas limit (6721975), hardfork (MUIRGLACIER), network ID (5777), RPC server (HTTP://127.0.0.1:8545), and mining status (AUTOMINING). A search bar for block numbers or tx hashes is also present. The main area shows a table of accounts with their addresses, balances, transaction counts, and indices. The first account has a balance of 999.99 ETH. The table includes columns for ADDRESS, BALANCE, TX COUNT, INDEX, and a key icon. The table has 6 rows. At the bottom, there is a status bar showing the weather (24°C, Clear), system icons, and the date (12-11-2022).

MNEMONIC	HD PATH
pumpkin consider advance elite knock grief monster body ritual man face sniff	m/44'/60'/0'/0/account_index
ADDRESS	
0xC1E88a5CF1082DD0674F09eA3CE9CFda3b264d20	BALANCE 999.99 ETH
ADDRESS	
0xd42B0Ecec2BCe68f76b240f8C6696779859f3137	BALANCE 999.85 ETH
ADDRESS	
0xD306dFD52eea4bFbD23587EBDBe8EA71f68C1c2B	BALANCE 1000.00 ETH
ADDRESS	
0x13503F6B21c35Cd58Db63AF38B834807e4543Bc5	BALANCE 1000.00 ETH
ADDRESS	
0x49955354711B1Fc26B45a76004aA590e95F90dEe	BALANCE 1000.00 ETH
ADDRESS	
0xd4dA436579cdc60339ab41f22E69f44332655401	BALANCE 1000.00 ETH



The screenshot shows the Remix - Ethereum IDE interface. At the top, there is a browser header with the URL remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.3+commit.8d00100cjs. The main area has a sidebar with sections for LEARNETH, 10.3 Transactions - Sending Ether, and examples for transfer, send, and call. The code editor on the right contains Solidity code for a Charity contract. The code defines a public state variable owner and a constructor that sets owner to msg.sender. It includes a payable function donate() and a withdraw() function that sends the contract's balance to the owner. A transaction log at the bottom shows a constructor call for a Charity contract with a value of 0 wei. The bottom status bar shows the weather (24°C, Clear), system icons, and the date (12-11-2022).

```
function sendViaCall(address payable _to) public payable {
    // Call returns a boolean value indicating success or failure.
    // This is the current recommended method to use.
    (bool sent, bytes memory data) = _to.call{value: msg.value}("");
    require(sent, "Failed to send Ether");
}

contract Charity {
    address public owner;
    constructor() {
        owner = msg.sender;
    }
    function donate()public payable {}
    function withdraw() public {
        uint amount = address(this).balance;
        (bool sent, bytes memory data) = owner.call{value: amount}("");
        require(sent, "Failed to send Ether");
    }
}
```