

ASSIGNMENT 06

1) Are Ommer blocks included in the blockchain?

Ans:

Ommer blocks are created in the Ethereum blockchain when two blocks are created and submitted to the ledger at roughly the same time.

Only one can enter the ledger. Ommer block generated another block after that.

2) What is CREATE2 in Solidity or EVM or Ethereum?

Ans: CREATE2 opcode gives us the ability predict the address where a contract will be deployed, without ever having to do so. This opens up lots of possibilities to improve user onboarding and scalability. In this guide we will precompute the address where a contract will be deployed and send Ether to it.

3) What happens if you try to store 256 using uint8?

ANS: it rolls over to zero after 255.

uint8 has a range of 0 to 2^{8-1}

4) What is the difference between address & address payable?

Ans: The address and address payable types both store a 160-bit Ethereum address.

The concept of payable and non-payable addresses only exists in the Solidity type system at compile-time. The difference between payable and non-payable addresses is gone in the compiled contract code.

5) Apart from msg.sender & block.timestamp, list all the global variables of Solidity.

Ans:

block.coinbase (address payable)	Current block miner's address
block.difficulty (uint)	Current block difficulty
msg.value (uint)	Number of wei sent with the message
block.number (uint):	Current block number
blockhash(uint blockNumber) returns (bytes32)	Gives hash of the given block and will only work for the 256 most recent block due to the reason of scalability.
block.timestamp:	Current block timestamp as seconds since unix epoch
gasleft() returns (uint256):	Remaining gas
msg.sender (address payable)	Sender of the message (current call)
msg.sig (bytes4)	First four bytes of the calldata (i.e. function identifier)
now (uint)	Current block timestamp (alias for block.timestamp)
tx.gasprice (uint)	Gas price of the transaction
block.gaslimit (uint)	Current block gaslimit
tx.origin (address payable)	Sender of the transaction (full call chain)
msg.data (bytes calldata)	Complete calldata

6) What is the difference between view & pure functions in Solidity?

Ans:

View functions are read only functions and do not modify the state of the block chain. In other words if you want to read data from the block chain one can use view.

Pure functions are more restrictive then view functions and do not modify the state *AND* do not read the state of the block chain.

	DEFAULT (NOT SPECIFIED IN FUNCTION)	VIEW	PURE
Ideal For	Transactions that alter data on the block chain	Getter functions to view data on the block chain	Defined to the scope of the function and do not alter or view data on the block chain
Data Access	read / write	read	None
Transaction Type	Send	Call	Call

7) Selfdestruct in Solidity?

Selfdestruct is a keyword that is used to terminate a contract, remove the bytecode from the Ethereum blockchain, and send any contract funds to a specified address.

8) Why memory keyword is used in case of arrays, strings & bytes in Solidity in function arguments or return statements?

Ans: Memory is a keyword used to store data for the execution of a contract. It holds functions argument data and is wiped after execution. storage can be seen as the default solidity data storage. It holds data persistently and consumes more gas.

9) How to call external functions from within the same contract in Solidity?

Ans: How do you call an external function in Solidity?

External functions are part of the contract interface, which means they can be called from other contracts and via transactions. An external function `f` cannot be called internally (i.e. `f()` does not work, but `this.f()` works). The two are incompatible, `public` allows internal calls, while `external` doesn't

***COMPLETE SOLIDITY BEGINNER COURSE FROM LEARNETH FROM REMIX IDE**

SOLIDITY BEGINNER COURSE SOLUTION CONTRACT SHOULD BE ADDED IN THE FOLLOWING ORDER

CHAPTER NAME

Assignment

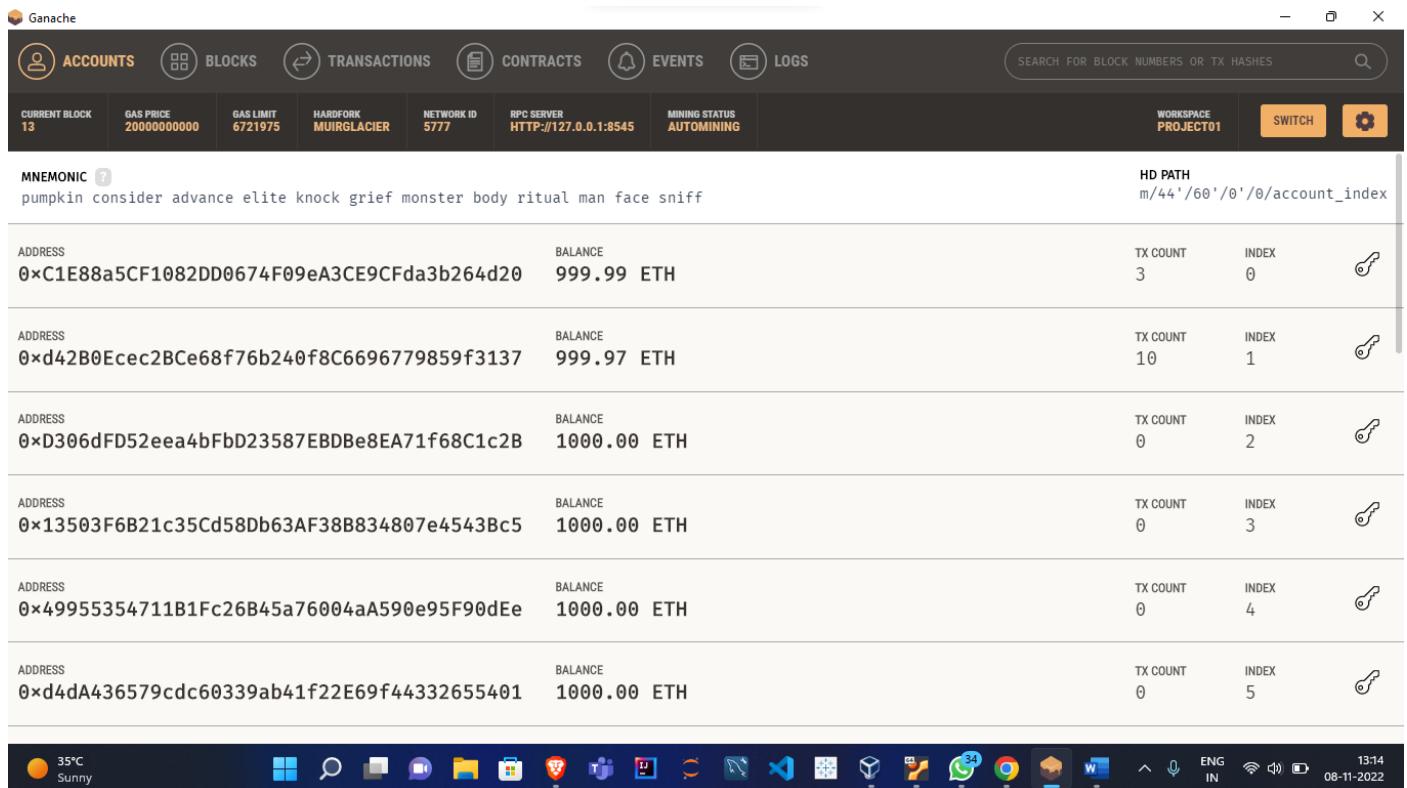
Details of the Assignment

SOLUTION CONTRACT [ENTIRE SMART CONTRACT CODE]

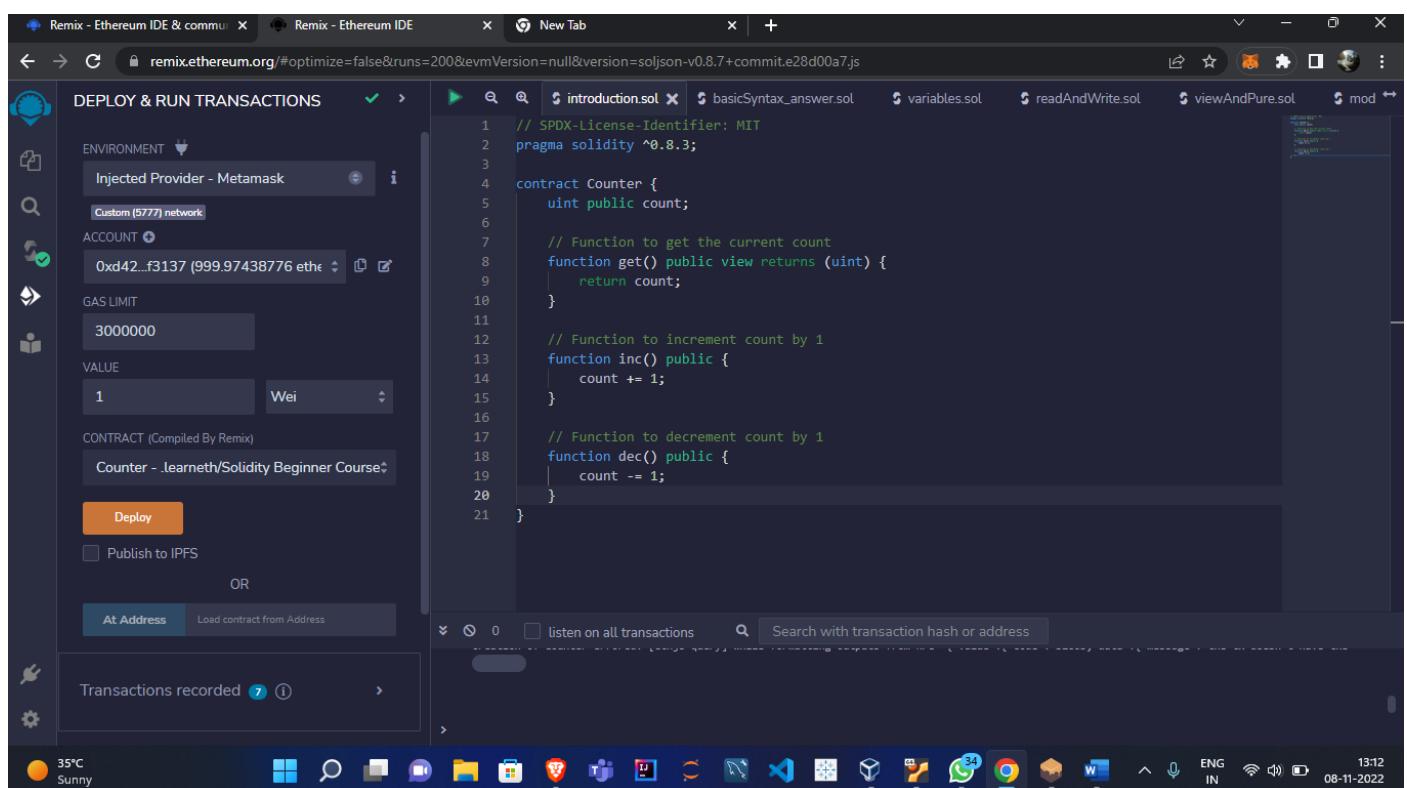
Deployed Address on Ganache

1. Introduction:

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



The screenshot shows the Ganache interface. At the top, there are tabs for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, and LOGS. Below the tabs, there are several status indicators: CURRENT BLOCK (13), GAS PRICE (20000000000), 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 mnemonic seed: "pumpkin consider advance elite knock grief monster body ritual man face sniff". Below this, a table lists six accounts with their addresses, balances, transaction counts, and indices. Each account row has a pencil icon for editing. The table columns are: ADDRESS, BALANCE, TX COUNT, INDEX, and a pencil icon. The accounts listed are: 0xC1E88a5CF1082DD0674F09eA3CE9CFda3b264d20 (999.99 ETH, 3 tx, index 0), 0xd42B0Ecec2BCe68f76b240f8C6696779859f3137 (999.97 ETH, 10 tx, index 1), 0xD306dFD52eea4bFbD23587EBDBe8EA71f68C1c2B (1000.00 ETH, 0 tx, index 2), 0x13503F6B21c35Cd58Db63AF38B834807e4543Bc5 (1000.00 ETH, 0 tx, index 3), 0x49955354711B1Fc26B45a76004aA590e95F90dEe (1000.00 ETH, 0 tx, index 4), and 0xd4dA436579cdc60339ab41f22E69f44332655401 (1000.00 ETH, 0 tx, index 5). The bottom of the interface shows a Windows taskbar with various icons and the date/time (08-11-2022, 13:14).



The screenshot shows the Remix Ethereum IDE interface. The top bar shows the URL: remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js. The main area is divided into two sections: "DEPLOY & RUN TRANSACTIONS" on the left and a code editor on the right. The code editor contains the following Solidity code for a Counter contract:

```
// 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;
    }

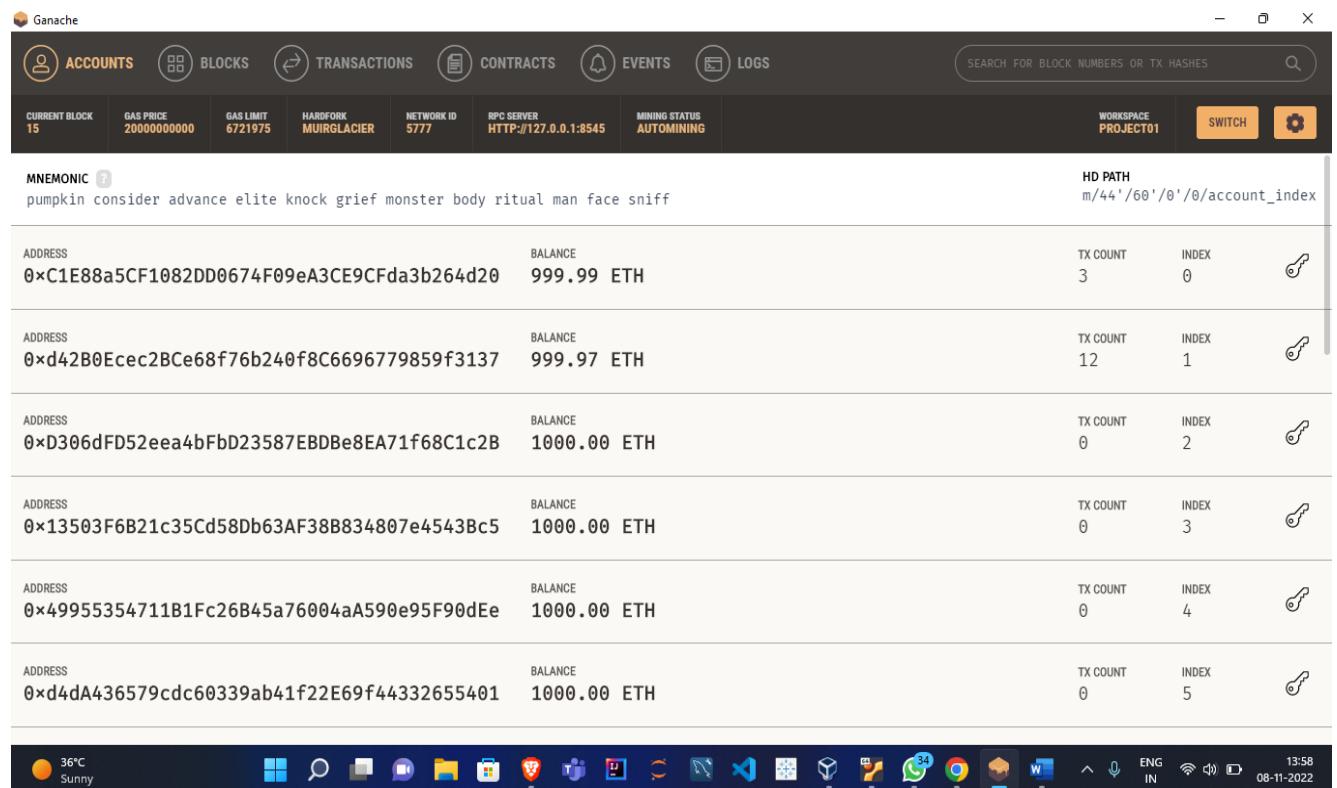
    // Function to increment count by 1
    function inc() public {
        count += 1;
    }

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

The "DEPLOY & RUN TRANSACTIONS" sidebar includes fields for ENVIRONMENT (Injected Provider - Metamask, Custom (5777) network), ACCOUNT (0xd42...f3137 (999.97438776 eth)), GAS LIMIT (300000), and VALUE (1 Wei). It also shows the CONTRACT (Counter) and provides a Deploy button. The bottom of the interface shows a Windows taskbar with various icons and the date/time (08-11-2022, 13:12).

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.



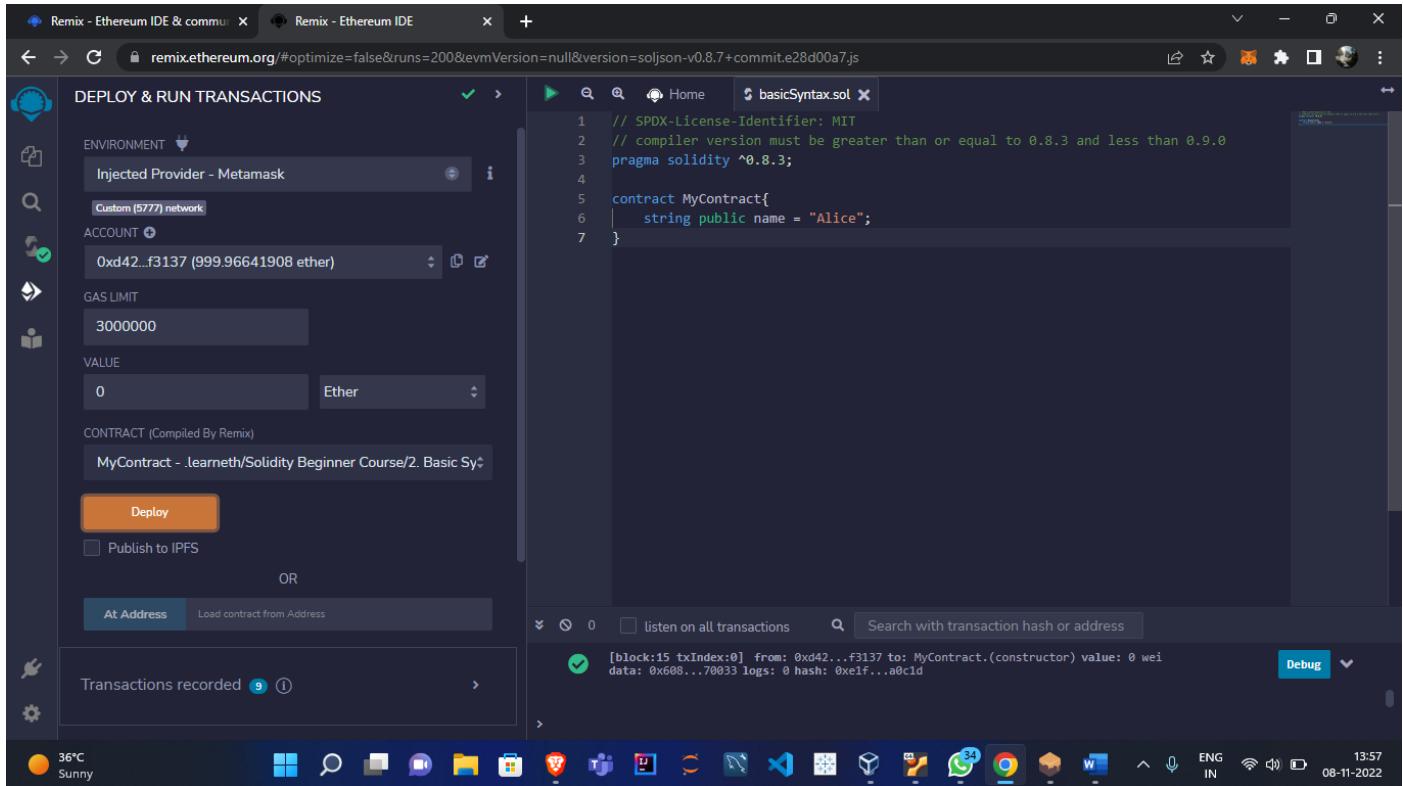
The screenshot shows the Ganache UI interface. At the top, there are tabs for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, and LOGS. Below the tabs, the current block is 15, gas price is 2000000000, gas limit is 6721975, the hardfork is MUIRGLACIER, network ID is 5777, and the RPC server is HTTP://127.0.0.1:8545. 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.

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	999.97 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	🔗

At the bottom, there is a taskbar with various icons including a weather widget (36°C Sunny), a search icon, and several application icons. The system tray shows the date (08-11-2022) and time (13:58).



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.

Ganache

ACCOUNTS BLOCKS TRANSACTIONS CONTRACTS EVENTS LOGS

SEARCH FOR BLOCK NUMBERS OR TX HASHES

CURRENT BLOCK 16 GAS PRICE 20000000000 GAS LIMIT 6721975 HARDFORK MUERGLACIER 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	999.96 ETH	13	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

28°C Sunny

Stop sharing Hide

17:02 08-11-2022

Remix - Ethereum IDE & commu x Remix - Ethereum IDE x Types — Solidity 0.8.18 document x (9430) Value Types | Solidity 0.8 - x +

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

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

1. 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`.
2. Create a `public` variable called `neg` that is a negative number, decide upon the type.
3. 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.

Check Answer Show answer

Stop sharing Hide

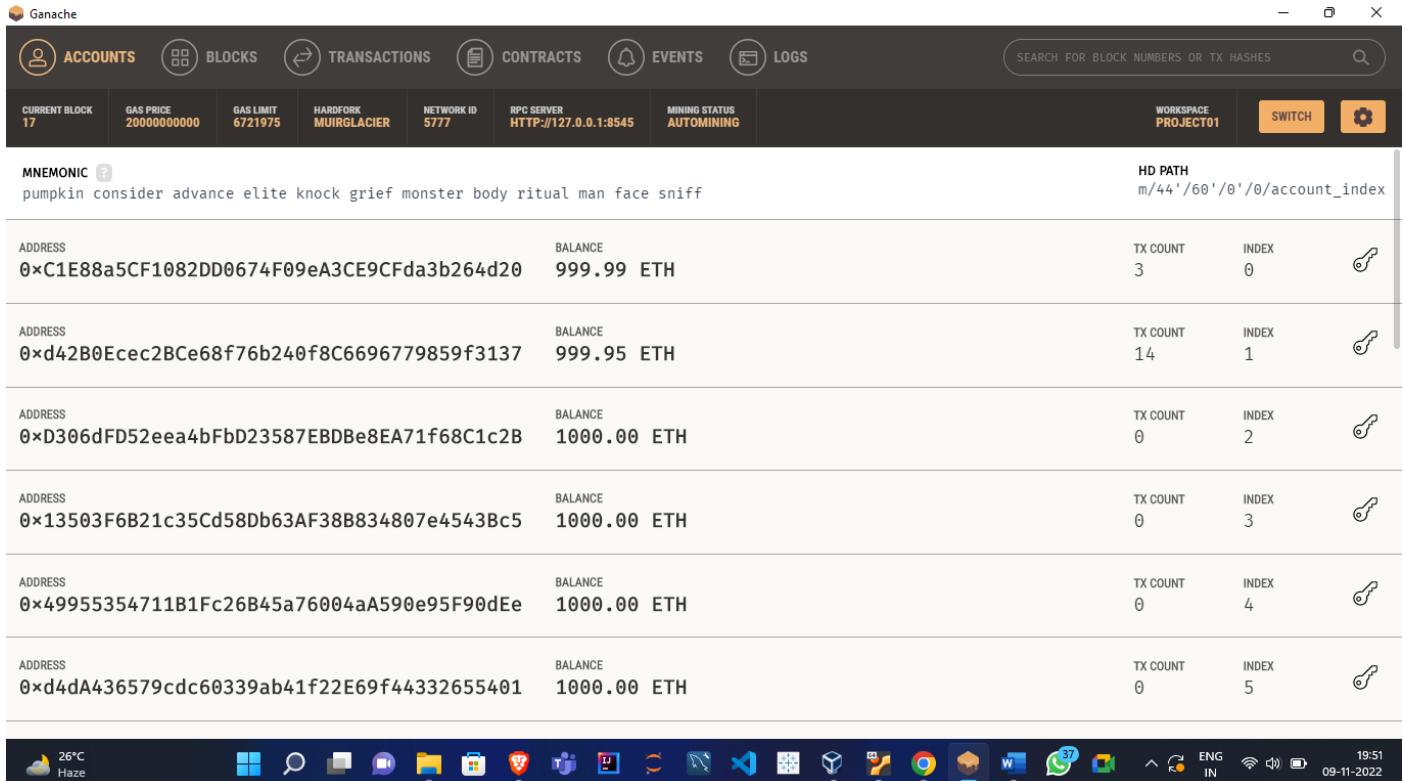
[block:16 txIndex:0] from: 0xd42...f3137 to: Primitives.(constructor) value: 0 wei data: 0xb08...0033 logs: 0 hash: 0xb2c...86ac5

17:03 08-11-2022

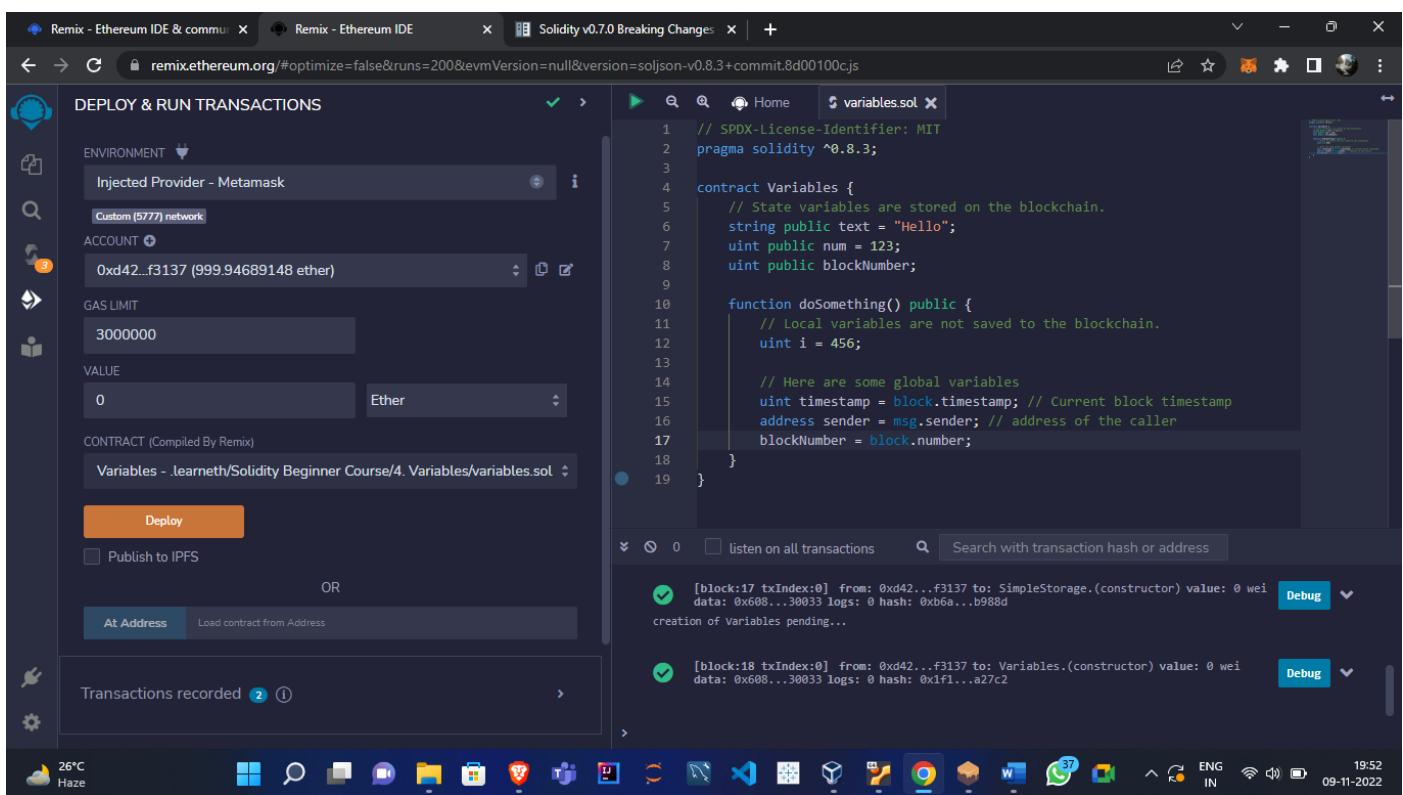
4. Variables : Assignment

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 interface. At the top, there are tabs for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, and LOGS. Below these are various status indicators: CURRENT BLOCK (17), 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. The main area displays a list of 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. The bottom of the screen shows a taskbar with various application icons.



The screenshot shows the Remix Ethereum IDE. On the left, the "DEPLOY & RUN TRANSACTIONS" sidebar shows the environment set to "Injected Provider - Metamask" (Custom (5777) network), account 0xd42...f3137 (999.94689148 ether), gas limit 3000000, and value 0 Ether. Below this, the "CONTRACT (Compiled By Remix)" section shows the contract name "Variables" and the file path ".learneth/Solidity Beginner Course/4. Variables/variables.sol". The main area displays the Solidity code for the "Variables" contract. The code includes a function `doSomething()` that is currently commented out. The Remix interface also shows transaction logs at the bottom, indicating the creation of the contract.

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 MUIRGLEACIER	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 0xC1E88a5CF1082DD0674F09eA3CE9CFda3b264d20	BALANCE 999.99 ETH				TX COUNT 3	INDEX 0
ADDRESS 0xd42B0Ecec2BCe68f76b240f8C6696779859f3137	BALANCE 999.95 ETH				TX COUNT 14	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

23°C Clear

Remix - Ethereum IDE & commu... 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 Custom (5777) network

ACCOUNT 0xd42...f3137 (999.95212308 ether)

GAS LIMIT 3000000

VALUE 0 Ether

CONTRACT (Compiled By Remix)

```
contract SimpleStorage {
    // State variable to store a number
    uint public num;
    bool public b = true;

    // You need to send a transaction to write to a state variable.
    function set(uint _num) public {
        num = _num;
    }

    // You can read from a state variable without sending a transaction.
    function get() public view returns (uint) {
        return num;
    }

    function get_b() public view returns (bool) {
        return b;
    }
}
```

Deploy Publish to IPFS

OR

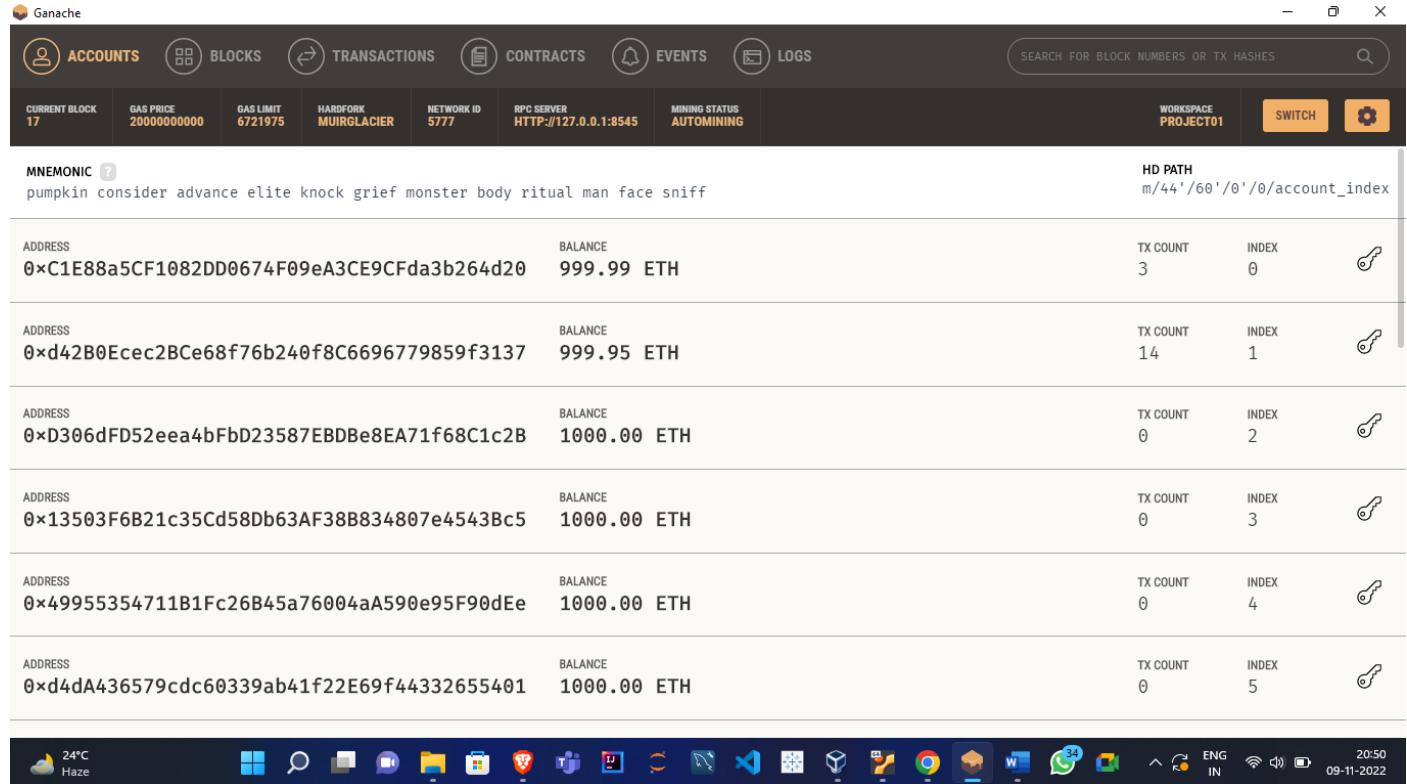
At Address Load contract from Address

Transactions recorded 1

23°C Clear

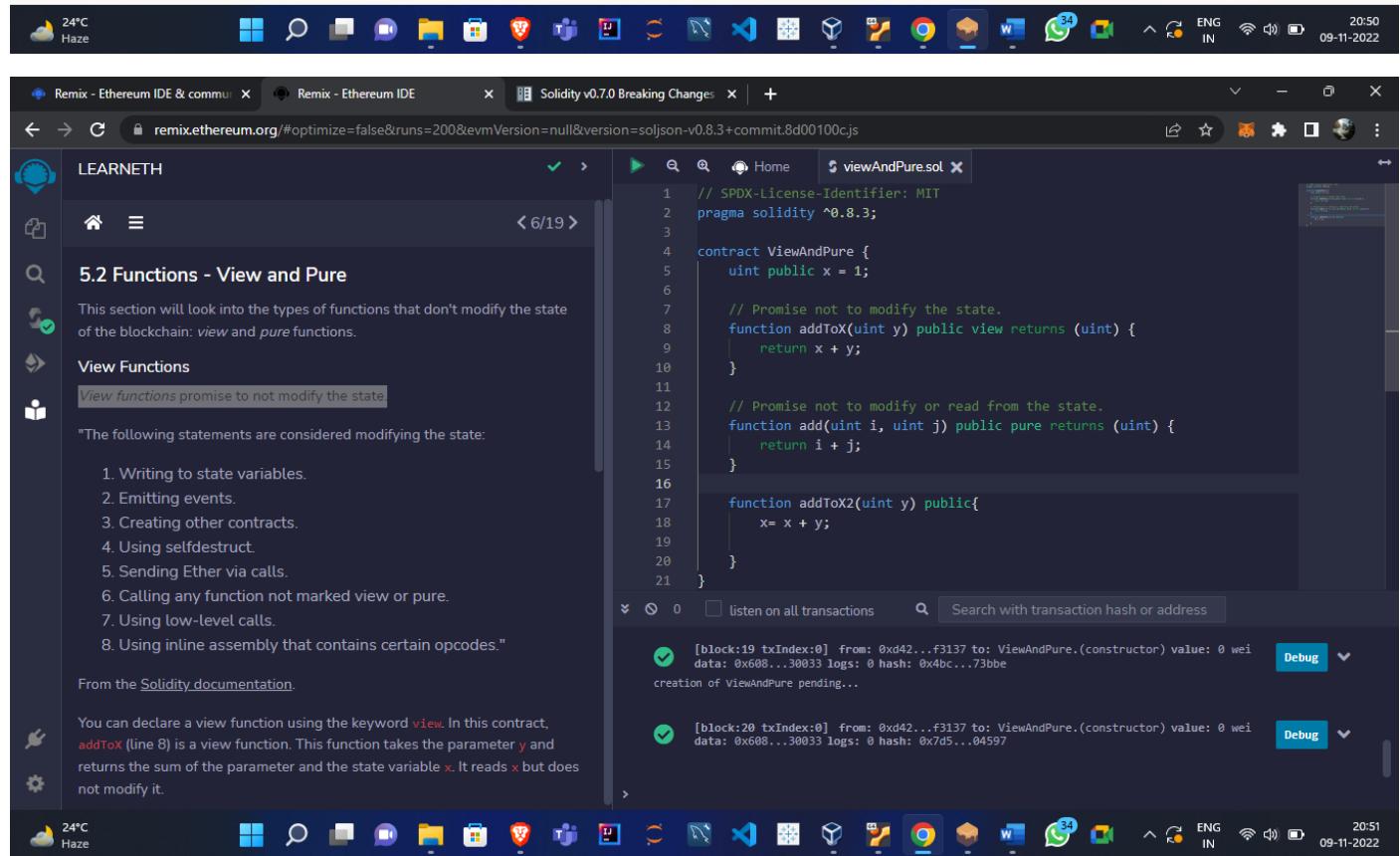
5.2) Functions - View and Pure : Assignment

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 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 (MUIRGLAGIER), 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 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, transaction counts, and indices. Each account row has a copy icon.

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	



The screenshot shows the Remix IDE interface. At the top, there are tabs for Remix - Ethereum IDE & community, Remix - Ethereum IDE, and Solidity v0.7.0 Breaking Changes. The URL is remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.3+commit.8d00100cjs. The bottom status bar shows the date as 09-11-2022 and the time as 20:50. The left sidebar shows a navigation tree with 'LEARNETH' selected, and a '5.2 Functions - View and Pure' section. This section discusses view and pure functions and lists statements that modify the state. The right panel shows the Solidity code for the 'ViewAndPure' contract. The code defines a state variable `x` and two functions: `addToX` (a view function that returns `x + y`) and `addToX2` (a pure function that updates `x` to `x + y`). Transaction logs are listed at the bottom, showing the creation of the contract and the execution of the `addToX2` function.

```
// 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;
    }

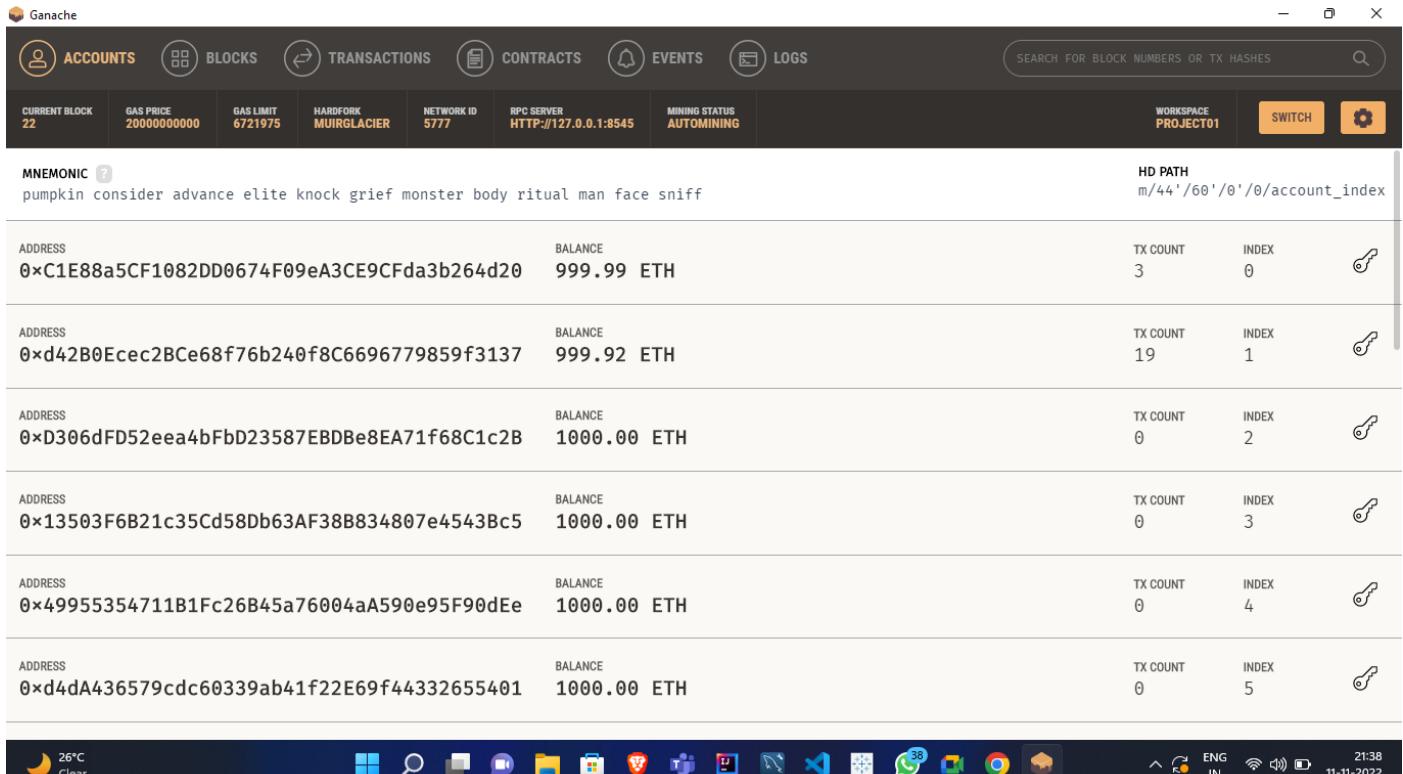
    function addToX2(uint y) public{
        x = x + y;
    }
}
```

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.

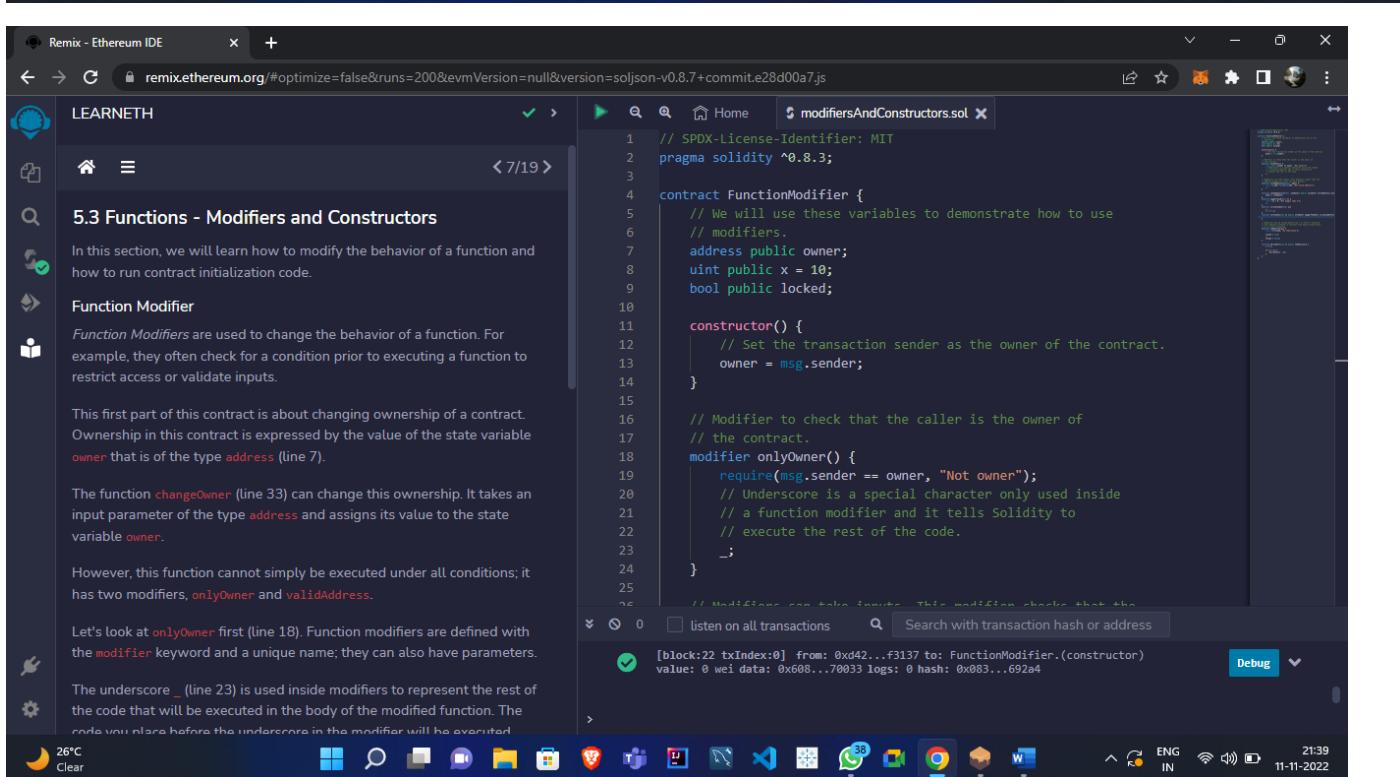
Make sure that `x` can only be increased.

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



The screenshot shows the Ganache interface with the following details:

ADDRESS	BALANCE	TX COUNT	INDEX	KEY
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	🔗



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

- Remix - Ethereum IDE** tab is active.
- The URL is `remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js`.
- LEARNETH** sidebar is open, showing the following content:

 - 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

- modifiersAndConstructors.sol** code editor:

```
// 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;

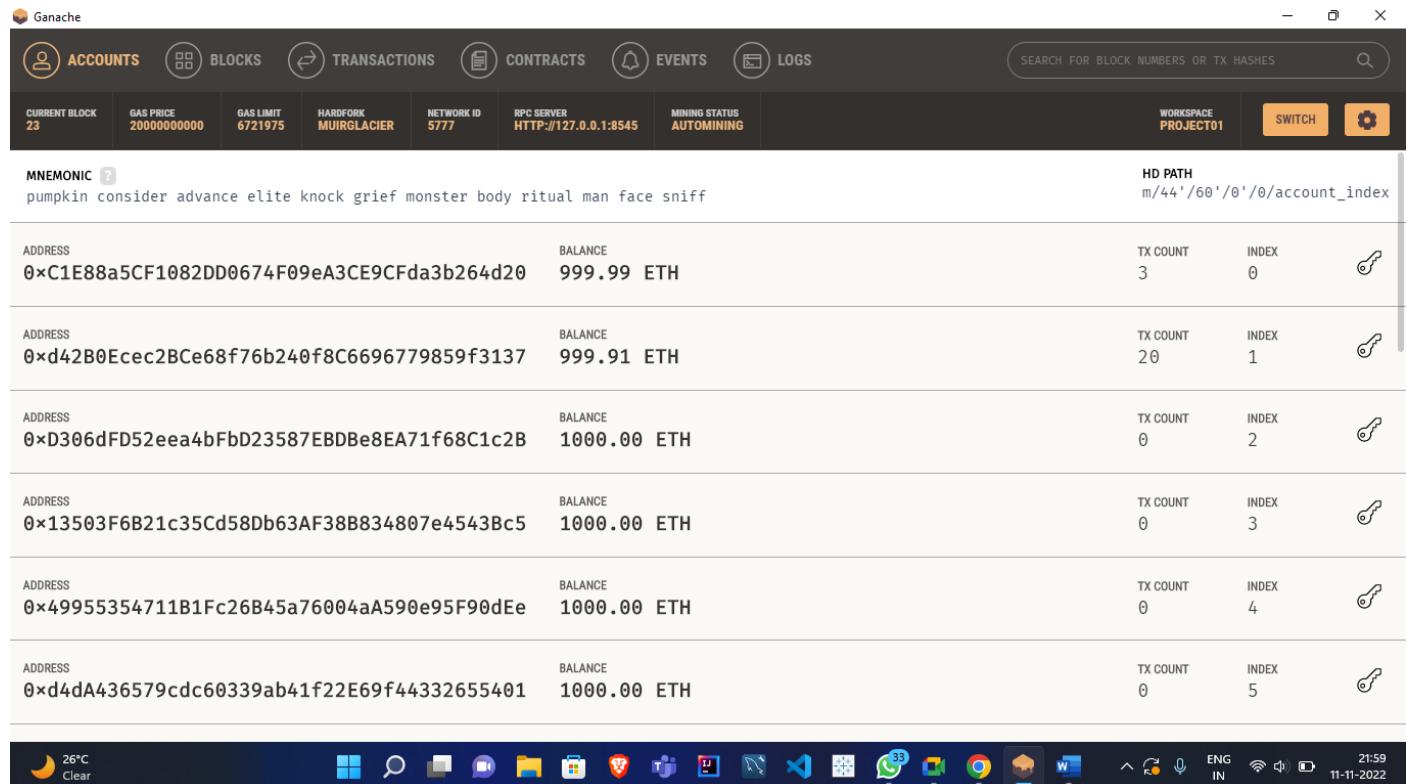
    constructor() {
        // Set the transaction sender as the owner of the contract.
        owner = msg.sender;
    }

    // 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.
        //
    }
}
```

- Bottom status bar shows: 26°C Clear, ENG IN, 21:38, 11-11-2022.

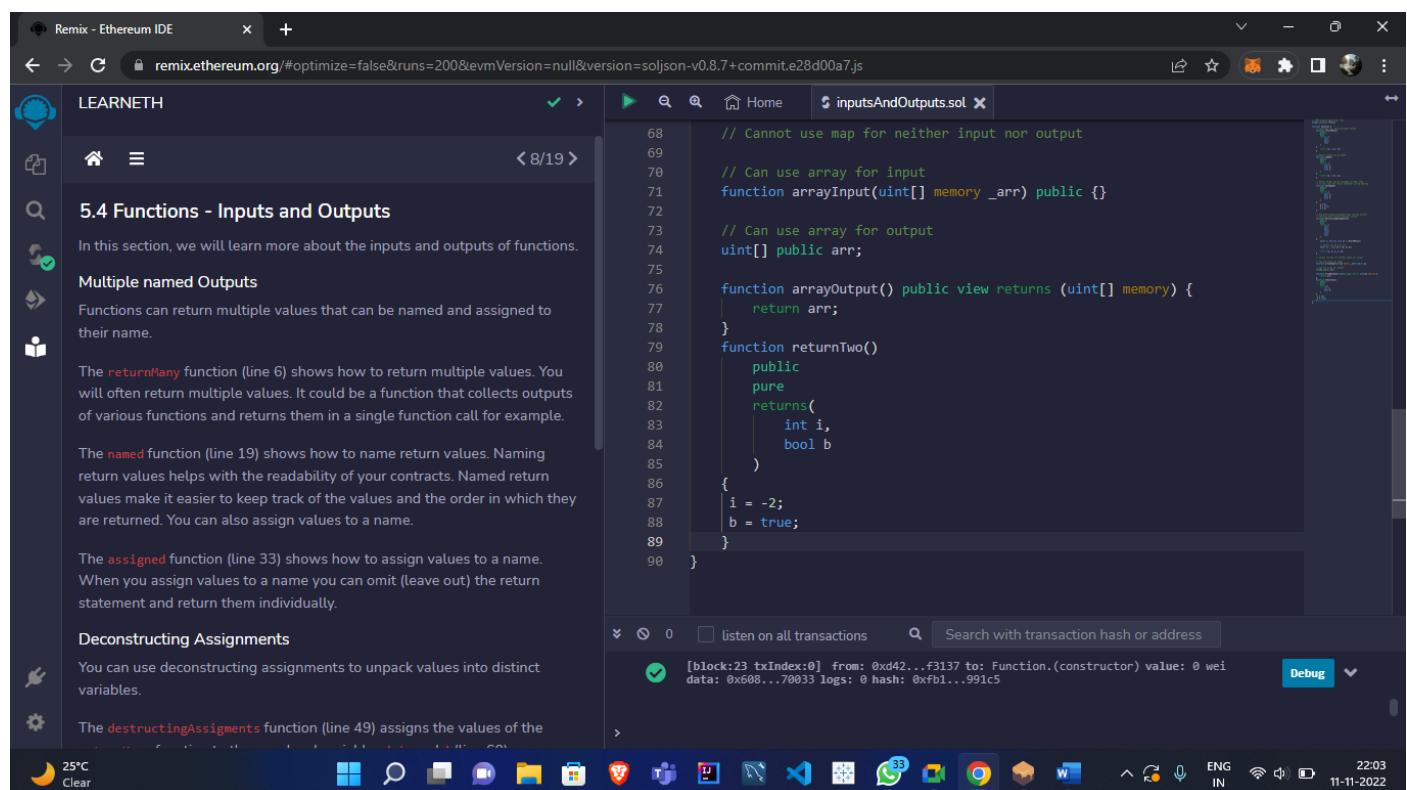
5.4 Functions - Inputs and Outputs :Assignment

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, the current block number is 23, gas price is 20000000000, gas limit is 6721975, the hardfork is MUIRGLACIER, the network ID is 5777, the RPC server is HTTP://127.0.0.1:8545, and the mining status is AUTOMINING. A workspace named PROJECT01 is selected. A search bar at the top right allows searching for block numbers or tx hashes. The main table lists accounts with their addresses, balances (e.g., 999.99 ETH, 999.91 ETH, 1000.00 ETH, 1000.00 ETH, 1000.00 ETH, 1000.00 ETH), transaction counts (e.g., 3, 20, 0, 0, 0, 0), and indices (e.g., 0, 1, 2, 3, 4, 5). A mnemonic seed "pumpkin consider advance elite knock grief monster body ritual man face sniff" is shown with its HD Path: m/44'/60'/0'/0/account_index. The bottom of the screen shows a Windows taskbar with various icons.

ADDRESS	BALANCE	TX COUNT	INDEX	
0xC1E88a5CF1082DD0674F09eA3CE9CFda3b264d20	999.99 ETH	3	0	🔑
0xd42B0Ecec2BCe68f76b240f8C6696779859f3137	999.91 ETH	20	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. The left sidebar has a sidebar menu with "LEARNETH" and a "5.4 Functions - Inputs and Outputs" section. The main area shows a Solidity code editor with the file "inputsAndOutputs.sol". The code defines a function `arrayInput` that takes an array of uints as input, and a function `arrayOutput` that returns an array of uints. It also defines a function `returnTwo` that returns -2 and true. The code editor includes line numbers and a sidebar with transaction logs and a debugger. The bottom of the screen shows a Windows taskbar with various icons.

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

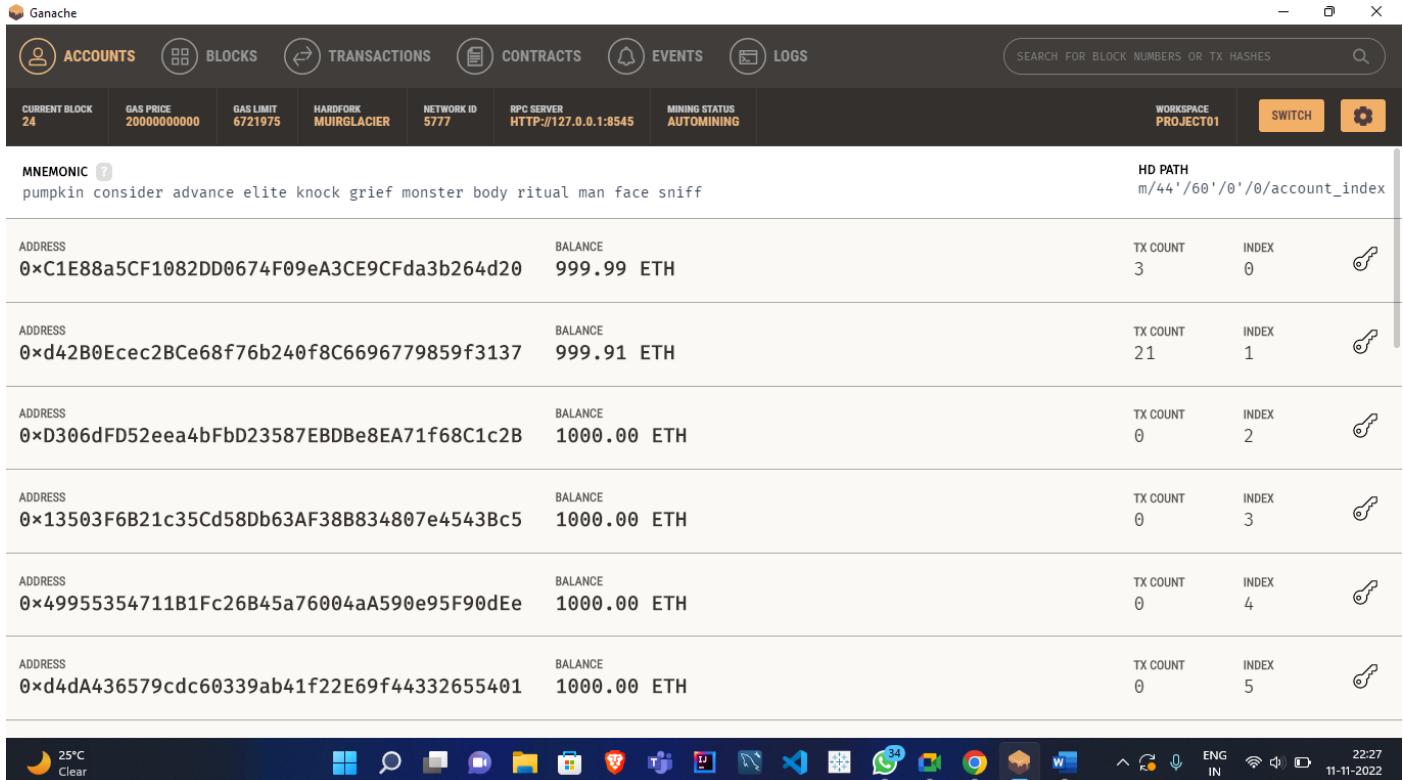
// Can use array for output
uint[] public arr;

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

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

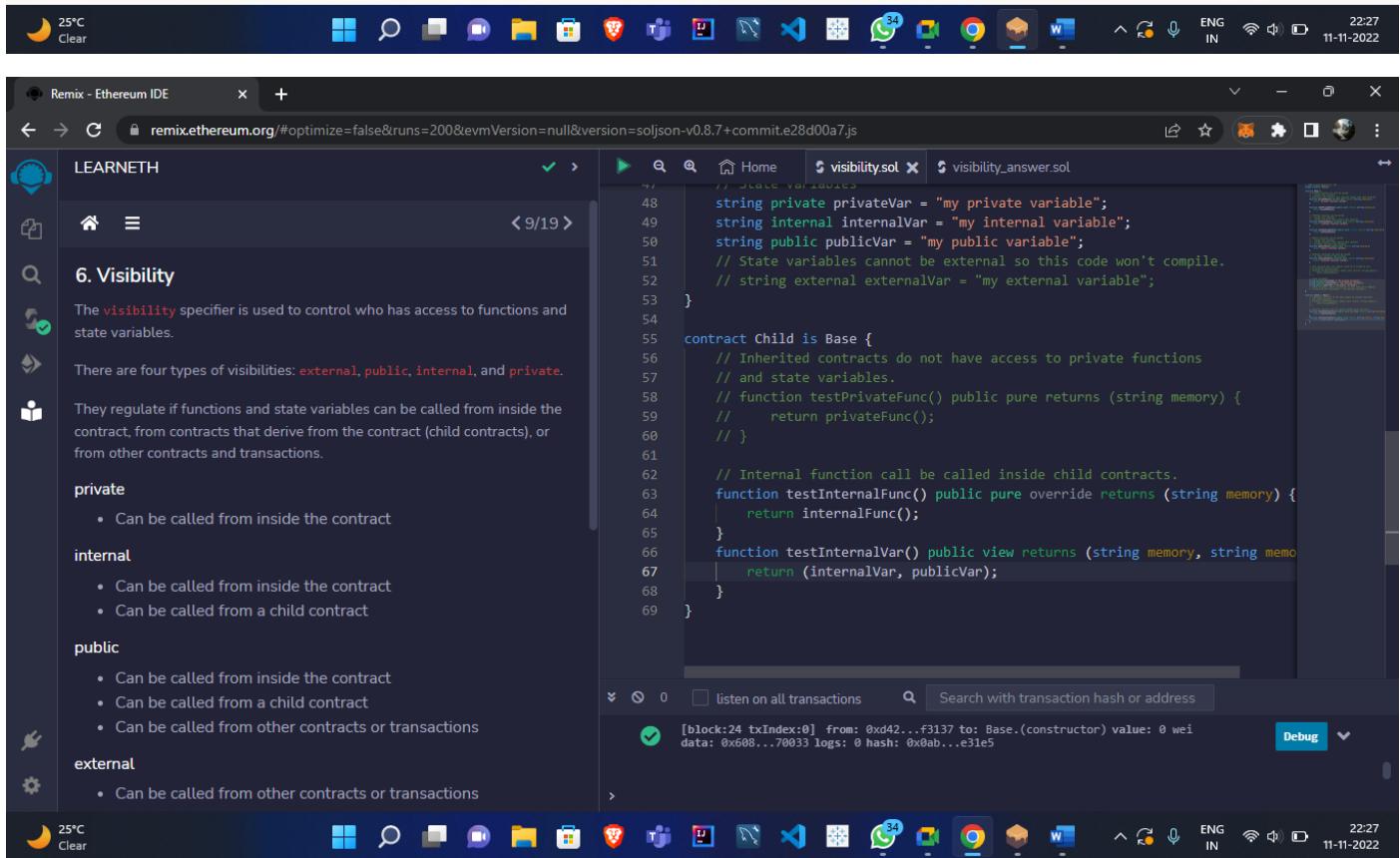
6. Visibility : Assignment

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, there are several status indicators: 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 displays a list of accounts with their addresses, balances, transaction counts, and indices. The first account is 0xC1E88a5CF1082DD0674F09eA3CE9CFda3b264d20 with a balance of 999.99 ETH. The second account is 0xd42B0Ecec2BCe68f76b240f8C6696779859f3137 with a balance of 999.91 ETH. The third account is 0xD306dFD52eea4bFbD23587EBDBe8EA71f68C1c2B with a balance of 1000.00 ETH. The fourth account is 0x13503F6B21c35Cd58Db63AF38B834807e4543Bc5 with a balance of 1000.00 ETH. The fifth account is 0x49955354711B1Fc26B45a76004aA590e95F90dEe with a balance of 1000.00 ETH. The sixth account is 0xd4dA436579cdc60339ab41f22E69f44332655401 with a balance of 1000.00 ETH.

MNEMONIC	HD PATH
pumpkin consider advance elite knock grief monster body ritual man face sniff	m/44'/60'/0'/0/account_index
ADDRESS	BALANCE
0xC1E88a5CF1082DD0674F09eA3CE9CFda3b264d20	999.99 ETH
ADDRESS	BALANCE
0xd42B0Ecec2BCe68f76b240f8C6696779859f3137	999.91 ETH
ADDRESS	BALANCE
0xD306dFD52eea4bFbD23587EBDBe8EA71f68C1c2B	1000.00 ETH
ADDRESS	BALANCE
0x13503F6B21c35Cd58Db63AF38B834807e4543Bc5	1000.00 ETH
ADDRESS	BALANCE
0x49955354711B1Fc26B45a76004aA590e95F90dEe	1000.00 ETH
ADDRESS	BALANCE
0xd4dA436579cdc60339ab41f22E69f44332655401	1000.00 ETH



The screenshot shows the Remix - Ethereum IDE interface. The top bar shows the URL `remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js`. The left sidebar has a sidebar title "LEARNETH" and a section "6. Visibility" which explains the `visibility` specifier. The right panel contains two code editors. The first code editor shows `visibility.sol` with the following code:`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";`The second code editor shows `visibility_answer.sol` with the following code:`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);
 }
}`At the bottom of the interface, there is a transaction log entry: "[block:24 txIndex:0] from: 0xd42...f3137 to: Base.(constructor) value: 0 wei data: 0x608...70033 logs: 0 hash: 0x0ab...e31e5".

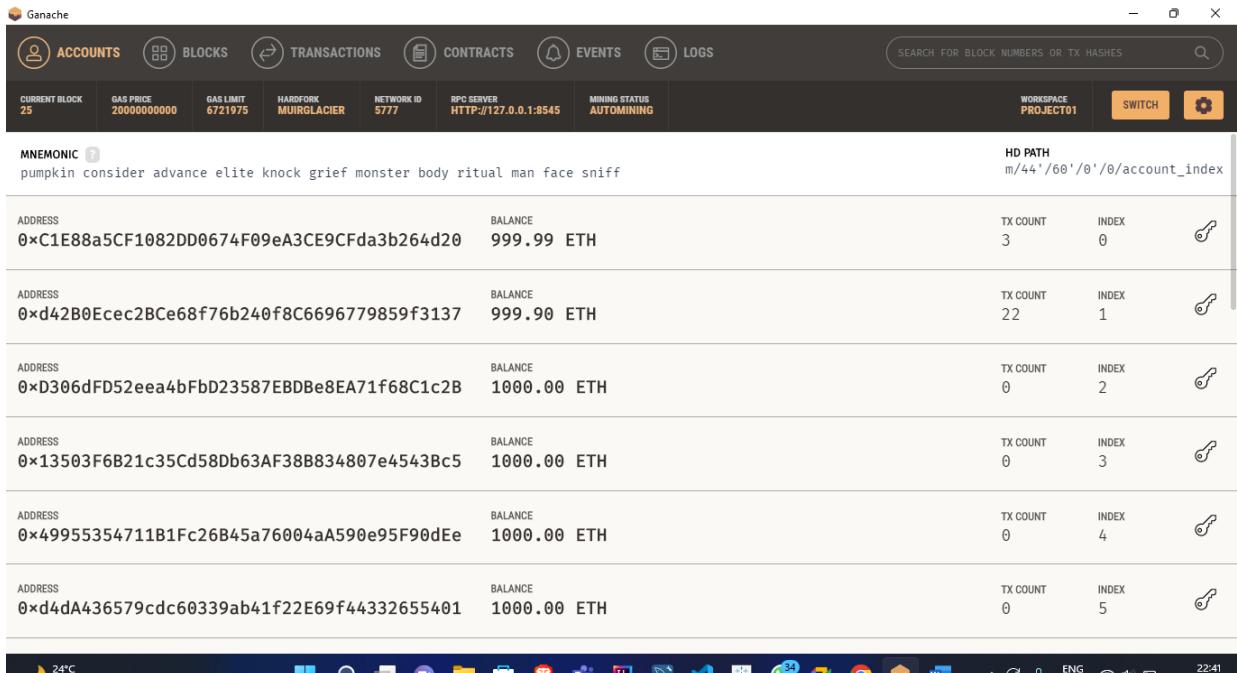
7.1 Control Flow - If/Else: Assignment

Create a new function called evenCheck in the IfElse contract:

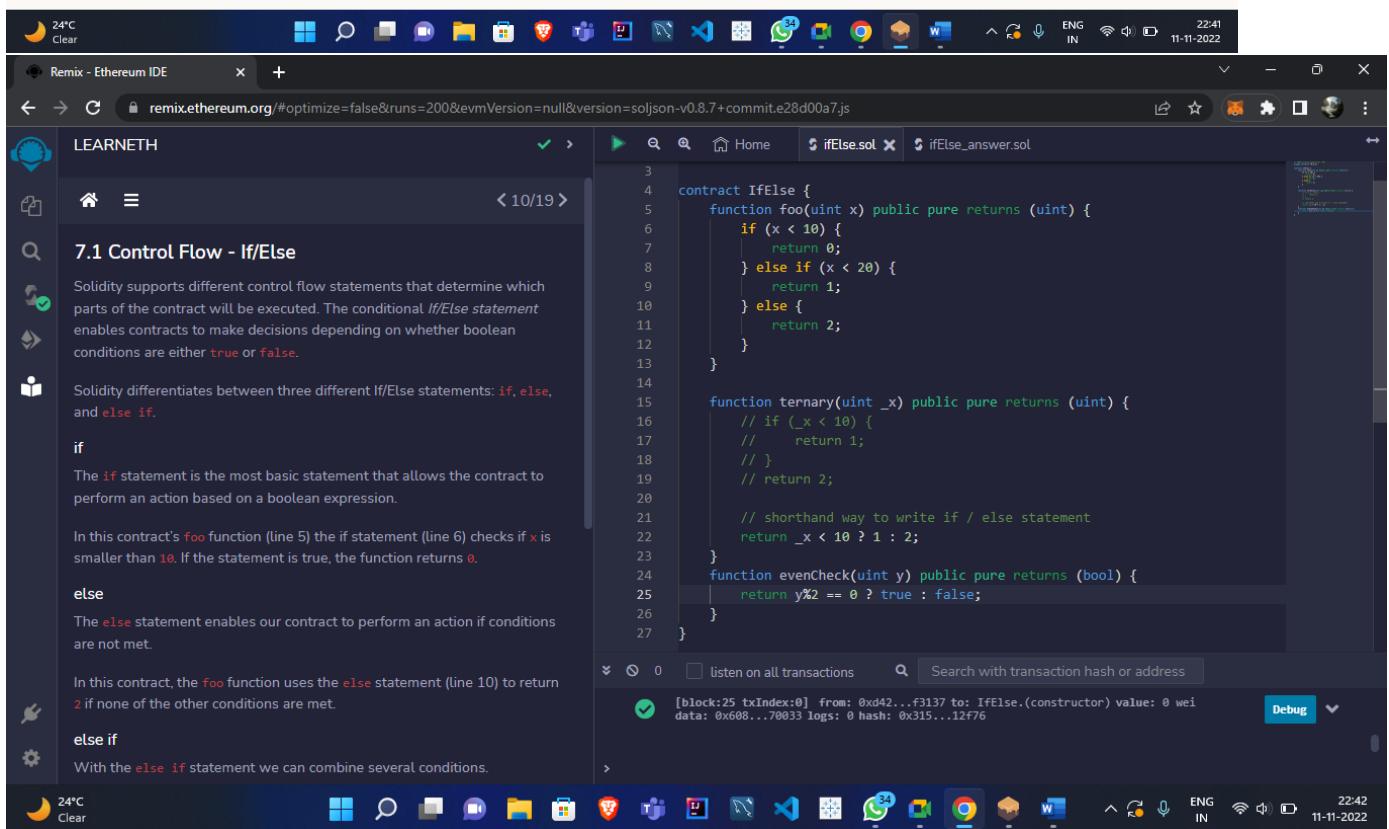
That takes in a uint as an argument.

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;
    }
}
```

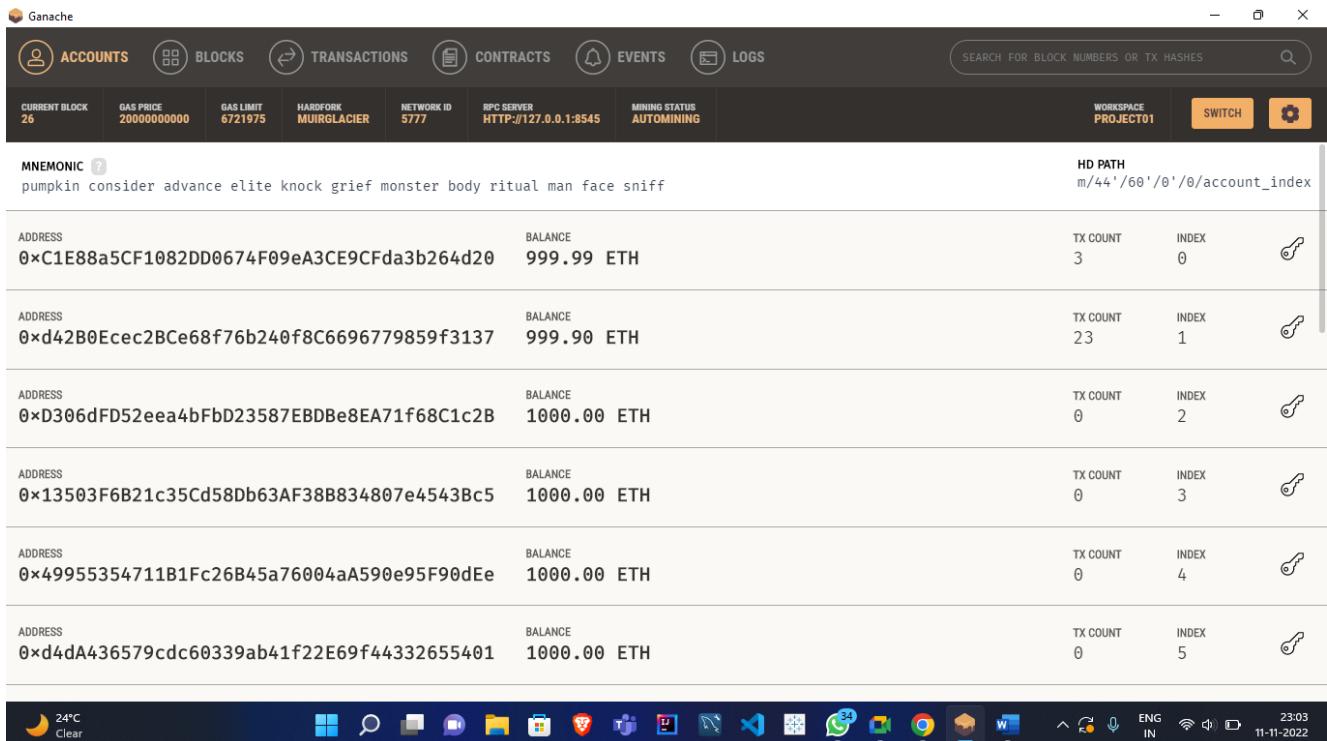
7.2 Control Flow – Loops: Assignment

Create a new function called evenCheck in the IfElse contract:

That takes in a uint as an argument.

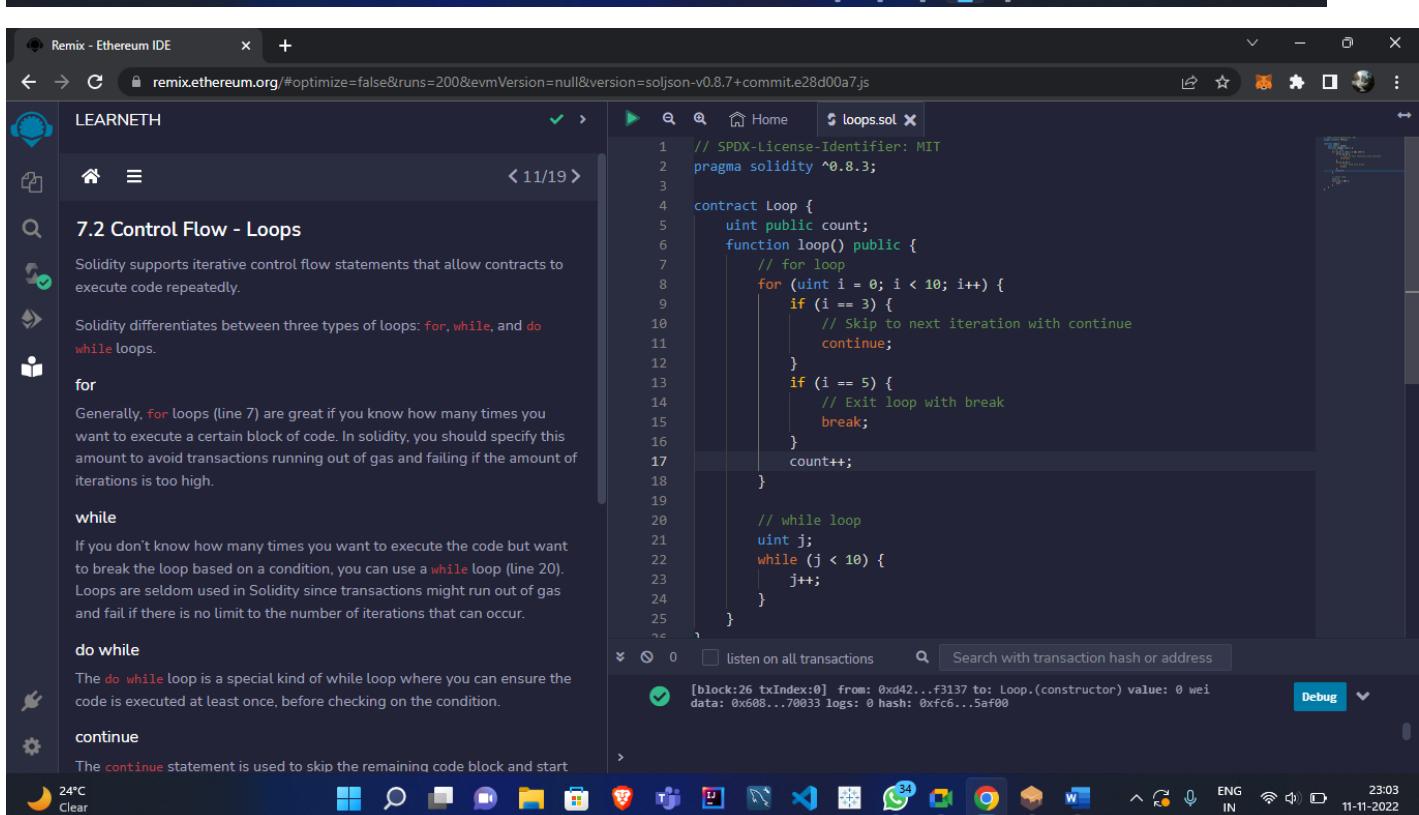
The function returns true if the argument is even, and false if the argument is odd.

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. On the right, there's a WORKSPACE PROJECT01, a SWITCH button, and a gear icon. A search bar at the top right says "SEARCH FOR BLOCK NUMBERS OR TX HASHES". Below the header is a table of accounts:

MNEMONIC	ADDRESS	BALANCE	TX COUNT	INDEX	KEY
pumpkin consider advance elite knock grief monster body ritual man face sniff	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	🔑



The screenshot shows the Remix Ethereum IDE. The top bar has tabs for "Remix - Ethereum IDE" and "loops.sol". The URL is "remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js". The sidebar on the left is titled "LEARNETH" and contains sections for "7.2 Control Flow - Loops", "for", "while", "do while", "continue", and "break". The main editor area contains the following Solidity 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++;
        }
    }
}
```

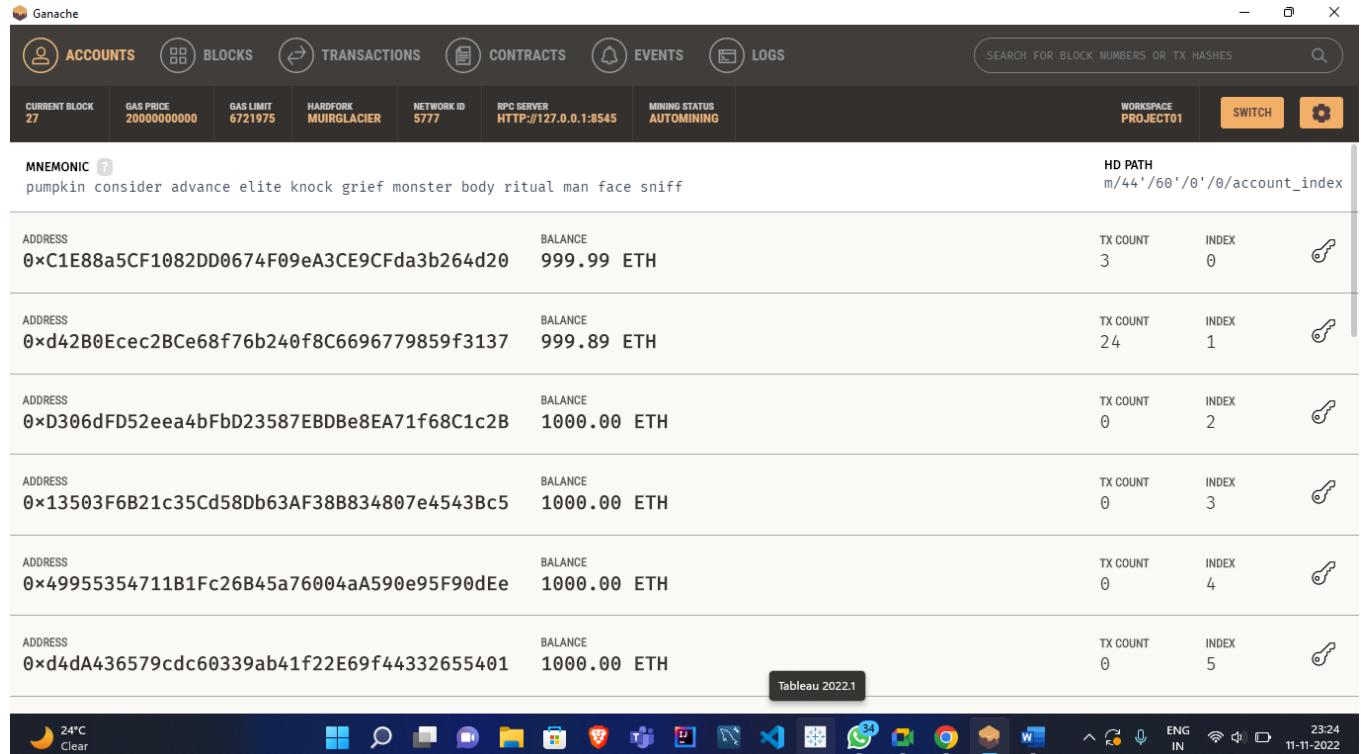
At the bottom, there's a transaction log: "[block:26 txIndex:0] from: 0xd42...f3137 to: Loop.(constructor) value: 0 wei data: 0x608...70033 logs: 0 hash: 0xfc6...5af00". The status bar shows "24°C Clear" and the date "11-11-2022".

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.

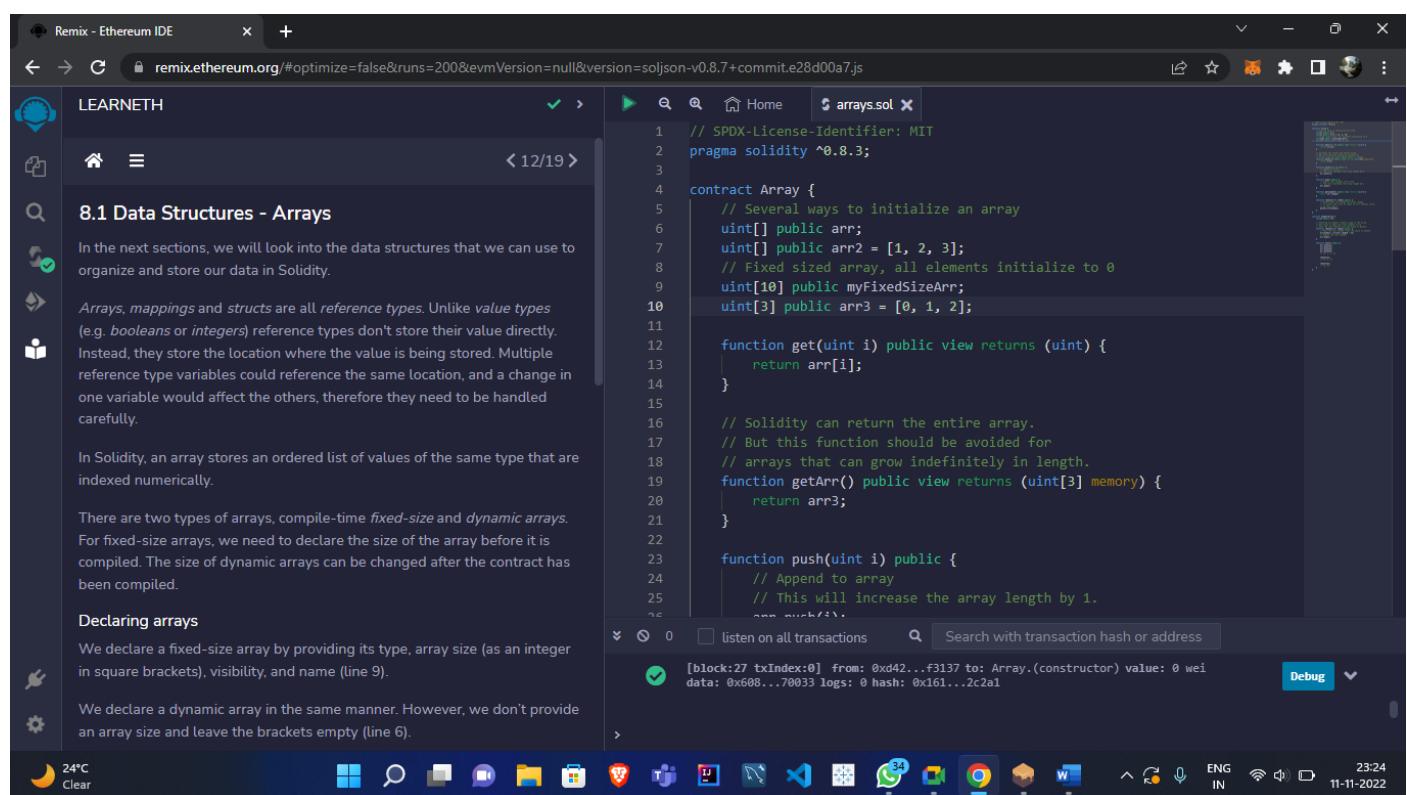
Change the getArr() function to return the value of arr3.



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 in Solidity. 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);
    }
}
```

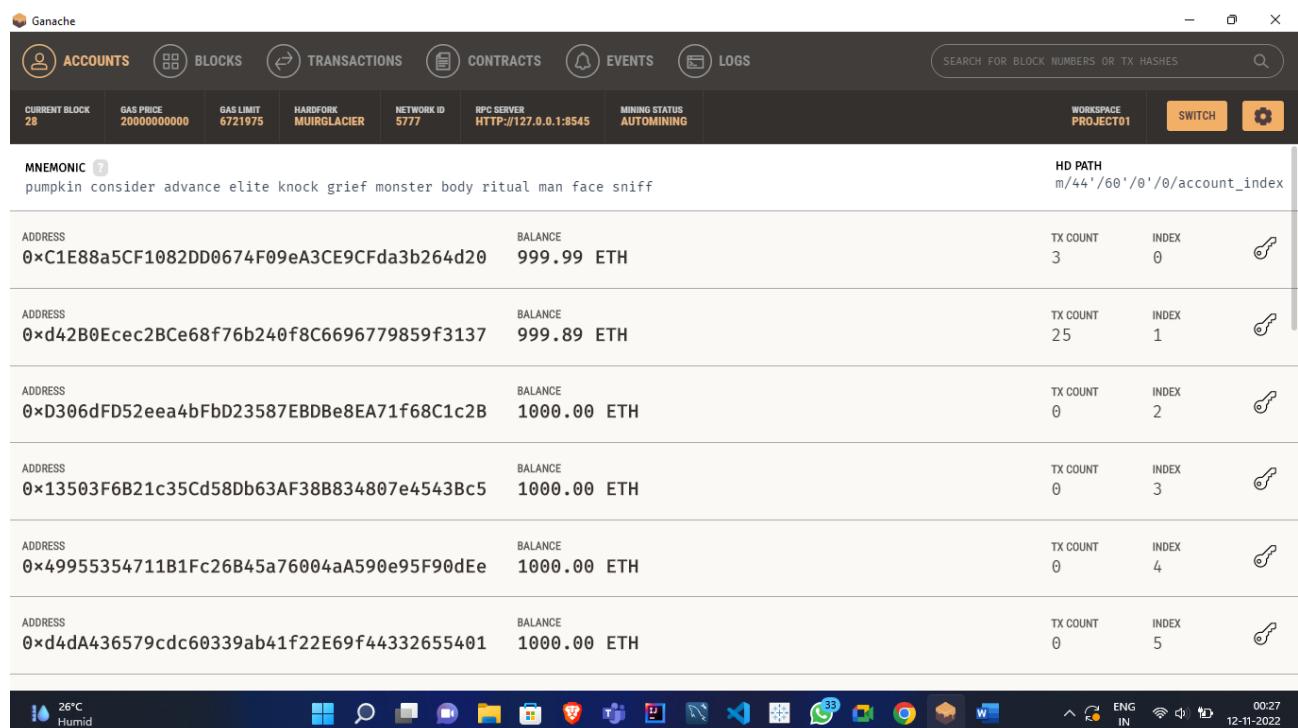
The status bar at the bottom shows the date and time as 23:24 11-11-2022.

8.2 Data Structures – Mappings: Assignment

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

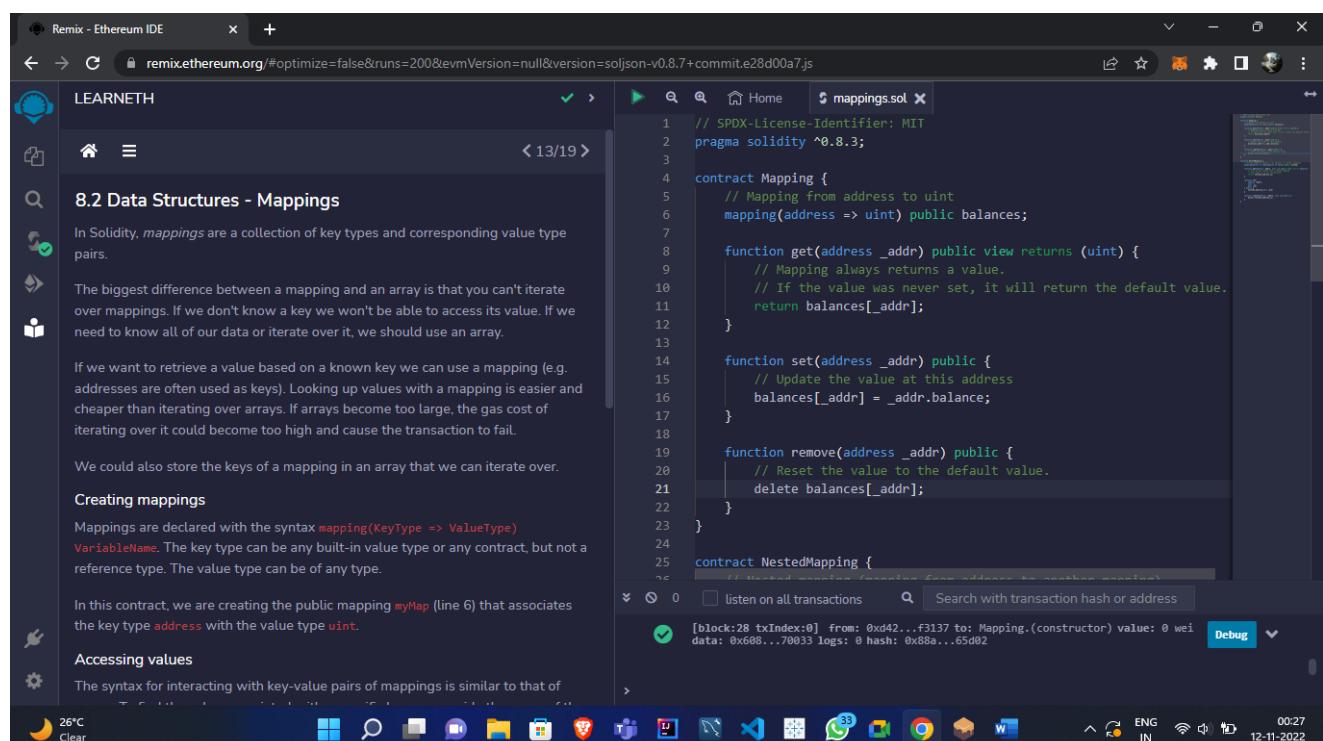
Change the functions get and remove to work with the mapping balances.

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 list of sections including "LEARNETH", "8.2 Data Structures - Mappings", and "Creating mappings".
- Right Sidebar:** Shows the 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];
    }

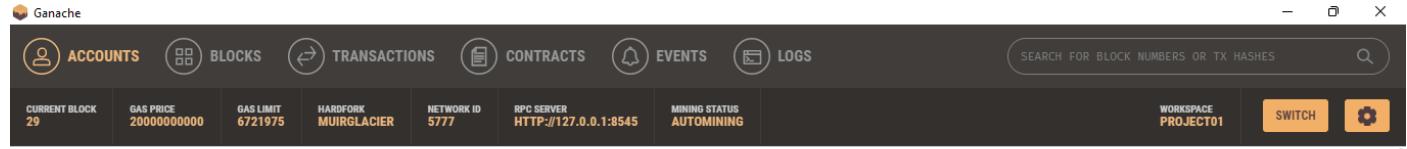
    function set(address _addr) public {
        // Update the value at this address
        balances[_addr] = _addr.balance;
    }

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

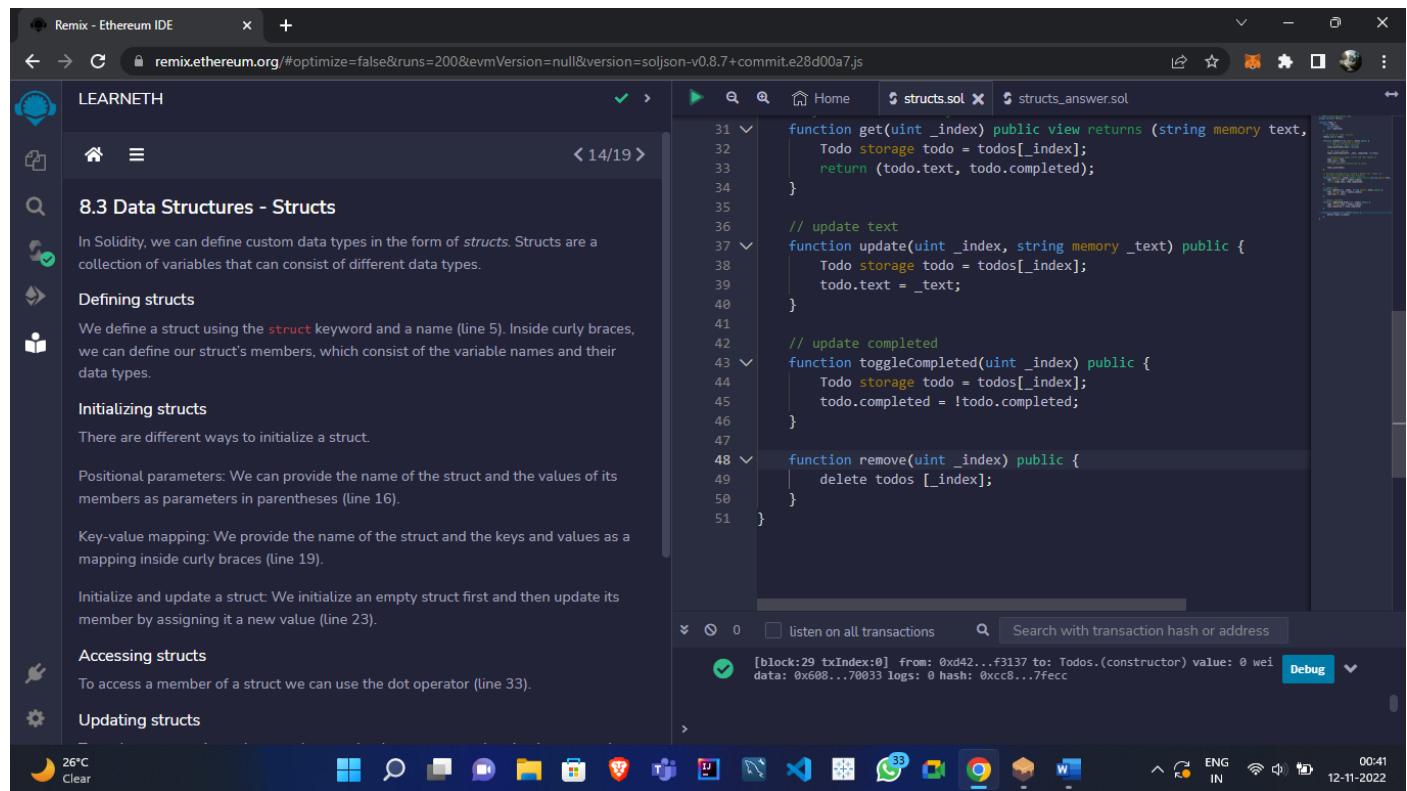
contract NestedMapping {
```
- Bottom Status Bar:** Shows the block number (28), transaction index (0), and a log entry: "[block:28 txIndex:0] from: 0xd42...f3137 to: Mapping.(constructor) value: 0 wei data: 0x608...70033 logs: 0 hash: 0x88a...65d02".

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.



Ganache											
ACCOUNTS	BLOCKS	TRANSACTIONS	CONTRACTS	EVENTS	LOGS	SEARCH FOR BLOCK NUMBERS OR TX HASHES					
CURRENT BLOCK 29	GAS PRICE 20000000000	GAS LIMIT 6721975	HARDFORK MUIRGLACIER	NETWORK ID 5777	RPC SERVER HTTP://127.0.0.1:8545	MINING STATUS AUTOMINING					
MNEMONIC						HD PATH m/44'/60'/0'/0/account_index					
pumpkin	consider	advance	elite	knock	grief	monster	body	ritual	man	face	sniff
ADDRESS 0xC1E88a5CF1082DD0674F09eA3CE9CFda3b264d20	BALANCE 999.99	ETH	TX COUNT 3	INDEX 0							
ADDRESS 0xd42B0Ecec2BCe68f76b240f8C6696779859f3137	BALANCE 999.87	ETH	TX COUNT 26	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							



Remix - Ethereum IDE 26°C Clear

remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7js

LEARNETH Home structs.sol structs_answer.sol

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

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

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

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

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

listen on all transactions Search with transaction hash or address

block:29 txIndex:0] from: 0xd42...f3137 to: Todos.(constructor) value: 0 wei Debug

data: 0x60...70033 logs: 0 hash: 0xcc8...7fecc

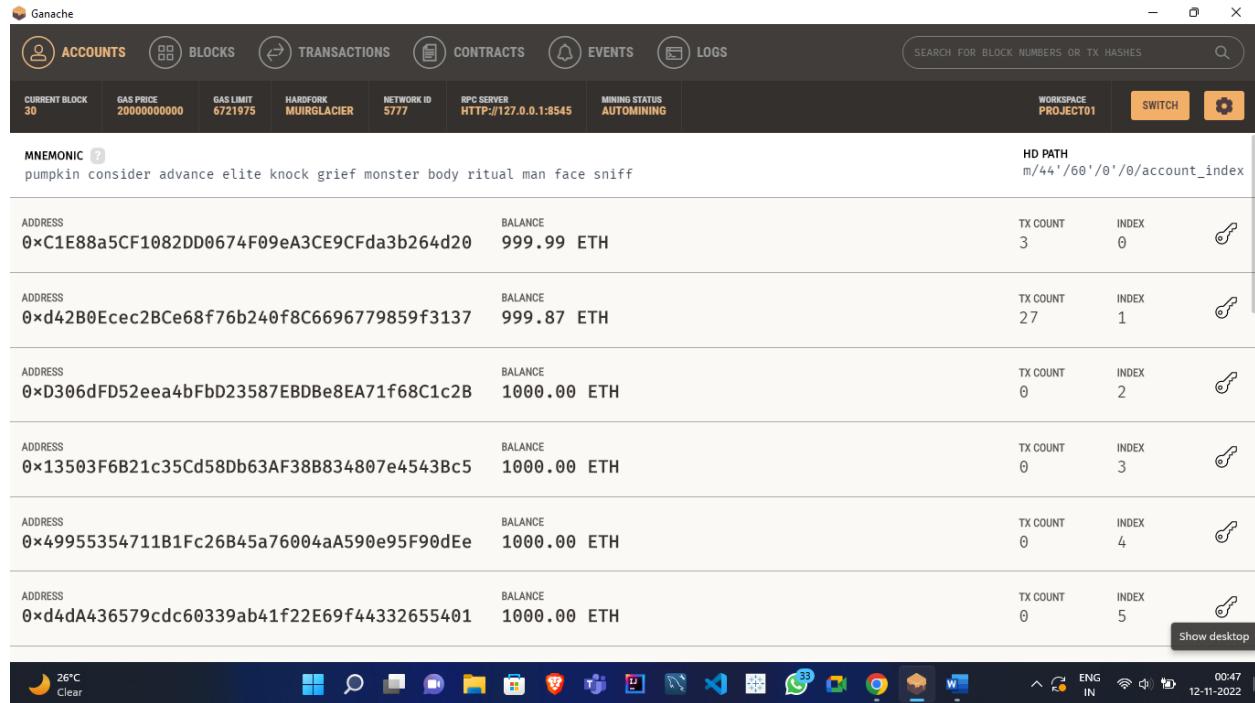
26°C Clear

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**.

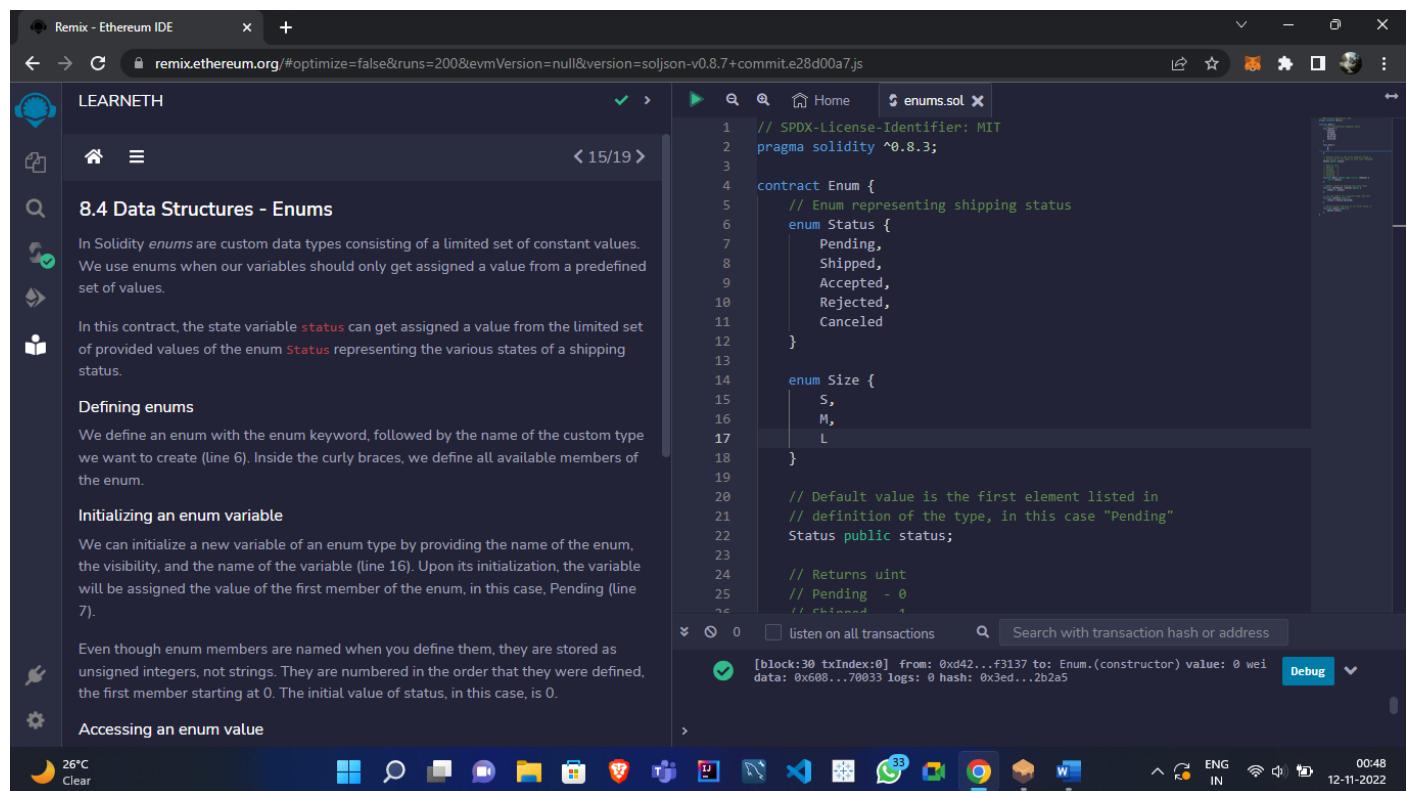
Create a getter function **getSize()** that returns the value of the variable sizes.



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

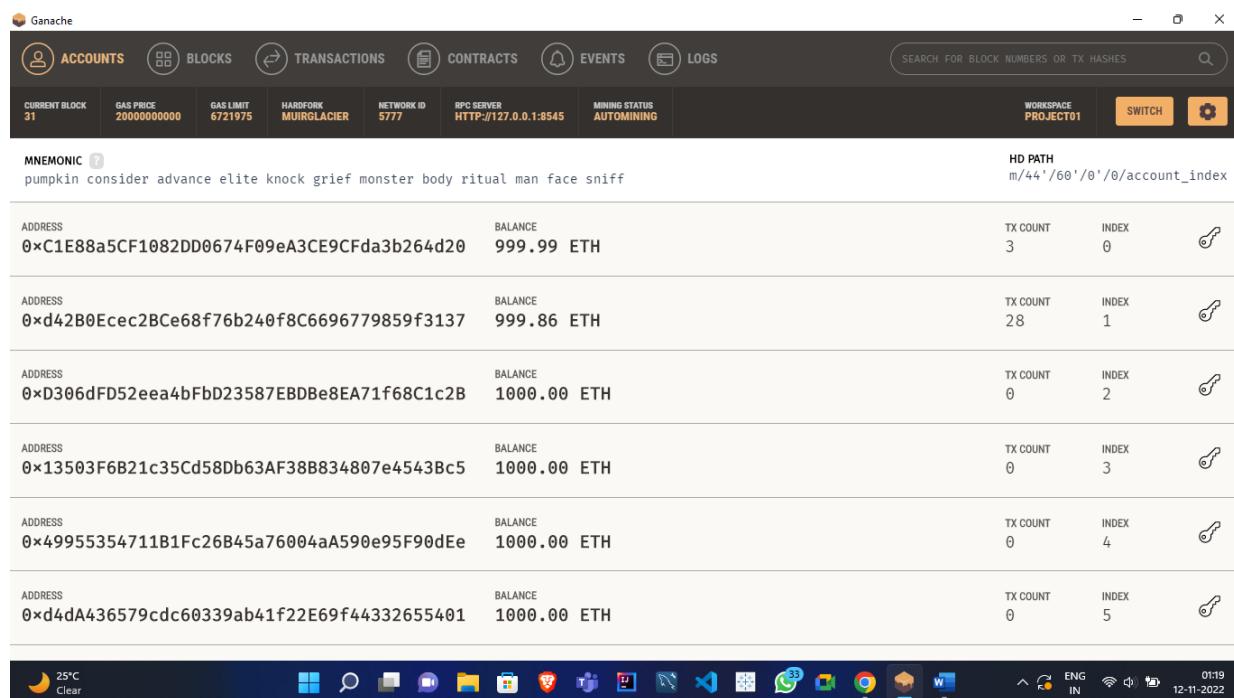
At the bottom, there is a "Show desktop" button and a weather widget showing 26°C and Clear. The taskbar at the bottom of the screen also shows various application icons.



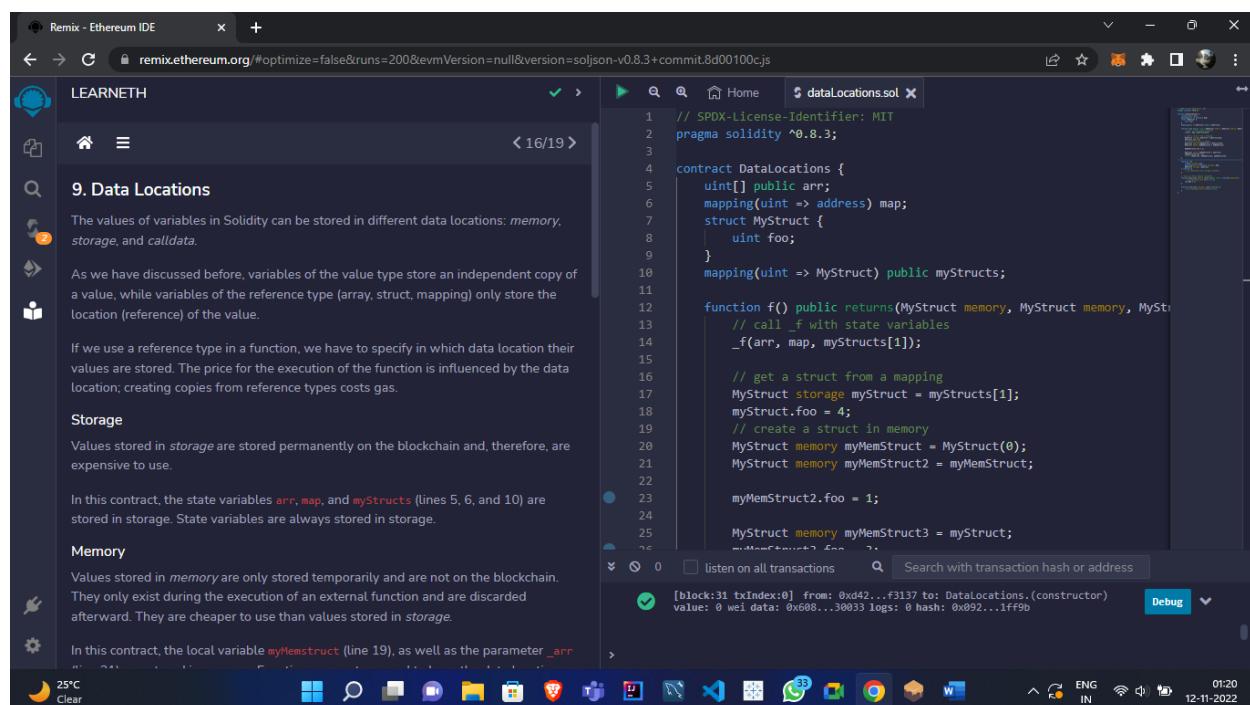
The screenshot shows the Remix - Ethereum IDE interface. The left sidebar has a "LEARNEETH" profile and a "8.4 Data Structures - Enums" section. The main area shows a Solidity code editor with the file "enums.sol". The code defines an enum **Status** with members **Pending**, **Shipped**, **Accepted**, **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 various network and mining status indicators. The mnemonic seed phrase is listed as: "pumpkin consider advance elite knock grief monster body ritual man face sniff". The HD PATH is shown as `m/44'/60'/0'/0/account_index`. The main table lists five Ethereum accounts with their addresses, balances (999.99, 999.86, 1000.00, 1000.00, 1000.00 ETH), transaction counts (3, 28, 0, 0, 0), and indices (0, 1, 2, 3, 4). Each account has a copy icon. The bottom of the interface shows a Windows taskbar with various icons and system status.



The screenshot shows the Remix Ethereum IDE. The left sidebar has a "LEARNETH" section with a "Data Locations" article. The main code editor shows the `dataLocations.sol` contract:`// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;

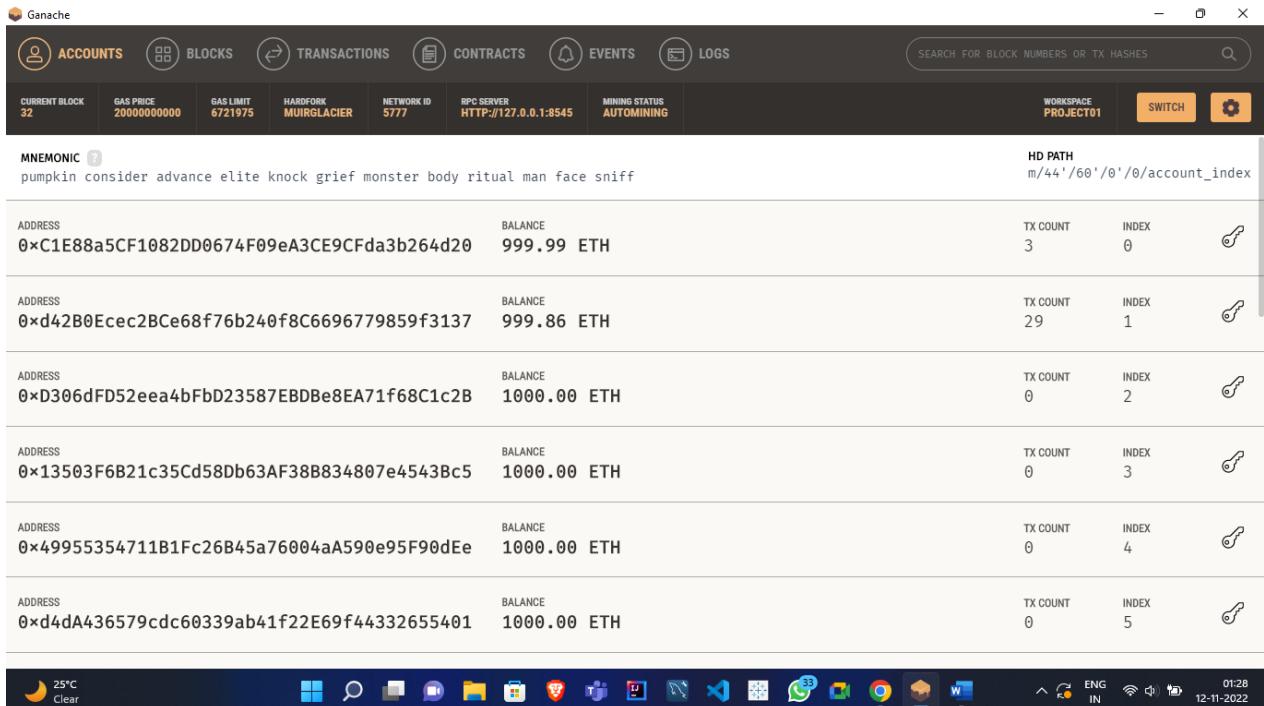
contract DataLocations {
 uint[] public arr;
 mapping(uint => address) map;
 struct MyStruct {
 uint foo;
 }
 mapping(uint => MyStruct) public myStructs;

 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 a transaction: [block:31 txIndex:0] from: 0x42...f3137 to: DataLocations.(constructor) value: 0 wei data: 0x0000...300033 logs: 0 hash: 0x892...1ff9b. The bottom of the interface shows a Windows taskbar with various icons and system status.

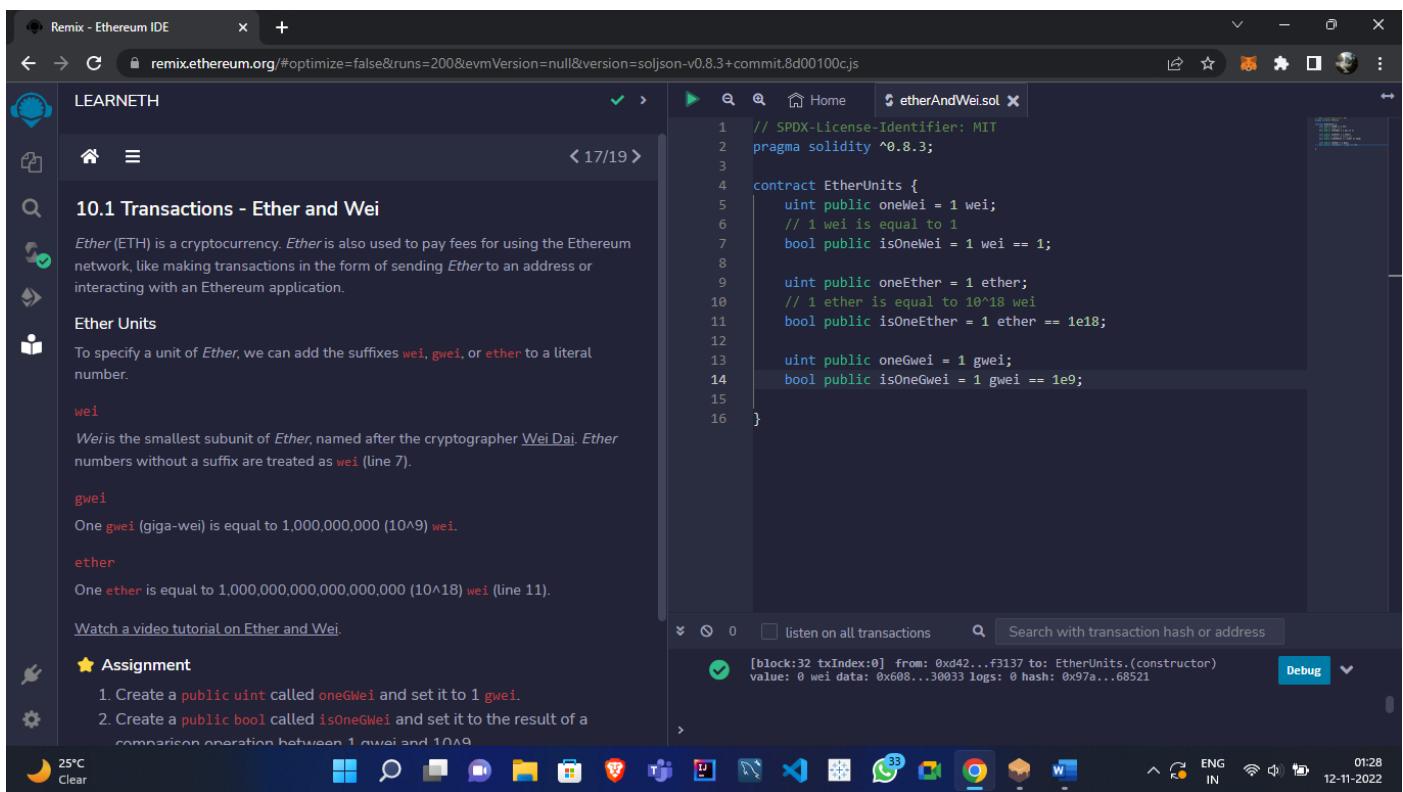
10.1 Transactions - Ether and Wei:

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 .



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 (32), GAS PRICE (20000000000), 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 transaction hashes. The main area displays a list of accounts with their addresses, balances, transaction counts, and indices. The first account has a balance of 999.99 ETH, 3 transactions, and index 0. The second account has a balance of 999.86 ETH, 29 transactions, and index 1. The third account has a balance of 1000.00 ETH, 0 transactions, and index 2. The fourth account has a balance of 1000.00 ETH, 0 transactions, and index 3. The fifth account has a balance of 1000.00 ETH, 0 transactions, and index 4. The sixth account has a balance of 1000.00 ETH, 0 transactions, and index 5. The bottom of the screen shows a Windows taskbar with various application icons.



The screenshot shows the Remix - Ethereum IDE interface. The left sidebar has a sidebar menu with sections like LEARNETH, Ether Units, and Assignment. The Ether Units section is expanded, showing definitions for `wei`, `gwei`, and `ether`. The Assignment section lists two tasks: 1. Create a `public uint` called `oneGWei` and set it to 1 `gwei`. 2. Create a `public bool` called `isOneGWei` and set it to the result of a comparison operation between 1 `gwei` and 10^9 . The right panel shows the Solidity code for the `EtherUnits` contract. The code includes SPDX license information, pragma solidity ^0.8.3, and a constructor that initializes `oneWei` to 1 `wei`, `oneEther` to 1 `ether`, and `oneGWei` to 1 `gwei`. It also includes a `isOneWei`, `isOneEther`, and `isOneGWei` function. The bottom of the screen shows a Windows taskbar with various application icons.

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

contract EtherUnits {
    uint public oneWei = 1 wei;
    // 1 wei is equal to 1
    bool public isOneWei = 1 wei == 1;

    uint public oneEther = 1 ether;
    // 1 ether is equal to 10^18 wei
    bool public isOneEther = 1 ether == 1e18;

    uint public oneGWei = 1 gwei;
    bool public isOneGWei = 1 gwei == 1e9;
}
```

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

ACCOUNTS BLOCKS TRANSACTIONS CONTRACTS EVENTS LOGS

SEARCH FOR BLOCK NUMBERS OR TX HASHES

CURRENT BLOCK 33 GAS PRICE 20000000000 GAS LIMIT 6721975 HARDFORK MURGLACIER 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	999.85 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

25°C Clear

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

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

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

0 listen on all transactions Search with transaction hash or address

Debug

25°C Clear

01:35 12-11-2022

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.

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
TX COUNT	3
INDEX	0
ADDRESS	
0xd42B0Ecec2BCe68f76b240f8C6696779859f3137	BALANCE 999.85 ETH
TX COUNT	31
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	
0x4995354711B1Fc26B45a76004aA590e95F90dEe	BALANCE 1000.00 ETH
TX COUNT	0
INDEX	4
ADDRESS	
0xd4dA436579cdc60339ab41f22E69f44332655401	BALANCE 1000.00 ETH
TX COUNT	0
INDEX	5

```
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");
    }
}
```