1. **Write a function to check whether a contract has code or not. Hint: Use assembly function.**

Answer:

CODE:

```
// SPDX-License-Identifier: MIT pragma solidity ^0.8.0;
contract ContractCodeChecker {
  // Function to check whether a contract has code at a given address function hasCode(address
  _target) external view returns (bool) {
    uint256 codeSize; assembly {
      // Retrieve the size of the code at the target address codeSize := extcodesize(_target)
    }
    return codeSize > 0;
  }
}
```

Explanation:

In the above code, I have defined a contract named ContractCodeChecker. The hasCode function takes a `_target` address as an argument and returns a boolean indicating whether the contract at that address has code or not. Inside the function, I have used assembly to retrieve the code size at the _target address using the `extcodesize` opcode. If the codeSize is greater than zero, the function returns true, indicating that the contract has code. Otherwise, it returns false.

1. **Explain advantages and disadvantages of using atomic swaps and cross-chain bridges.**

Atomic swaps and cross-chain bridges facilitate interoperability between blockchain networks, each with distinct pros and cons.

Atomic Swaps:

Advantages:

Decentralized and Trustless: Atomic swaps eliminate the need for a third-party intermediary, ensuring user control over private keys and funds.
Interoperability: They enable direct exchange between different blockchain networks, enhancing liquidity for lesser-known tokens.
Privacy: Transactions occur directly between parties, enhancing privacy compared to centralized exchanges.
Disadvantages:

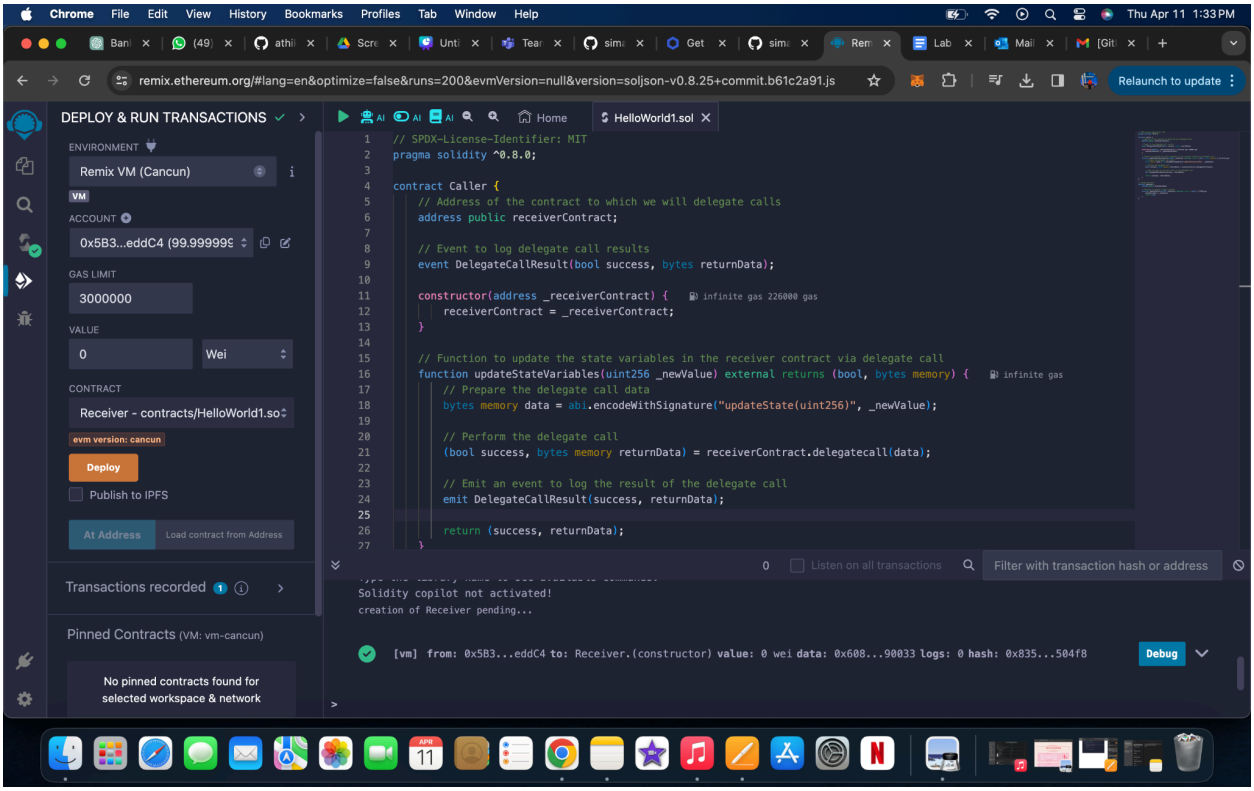
Complexity: Executing atomic swaps can be technically challenging for average users.
Liquidity: Finding suitable counterparties for swaps, especially for less popular tokens, can be difficult.
Limited Compatibility: Both blockchains involved must support the same hashing algorithm and scripting language.

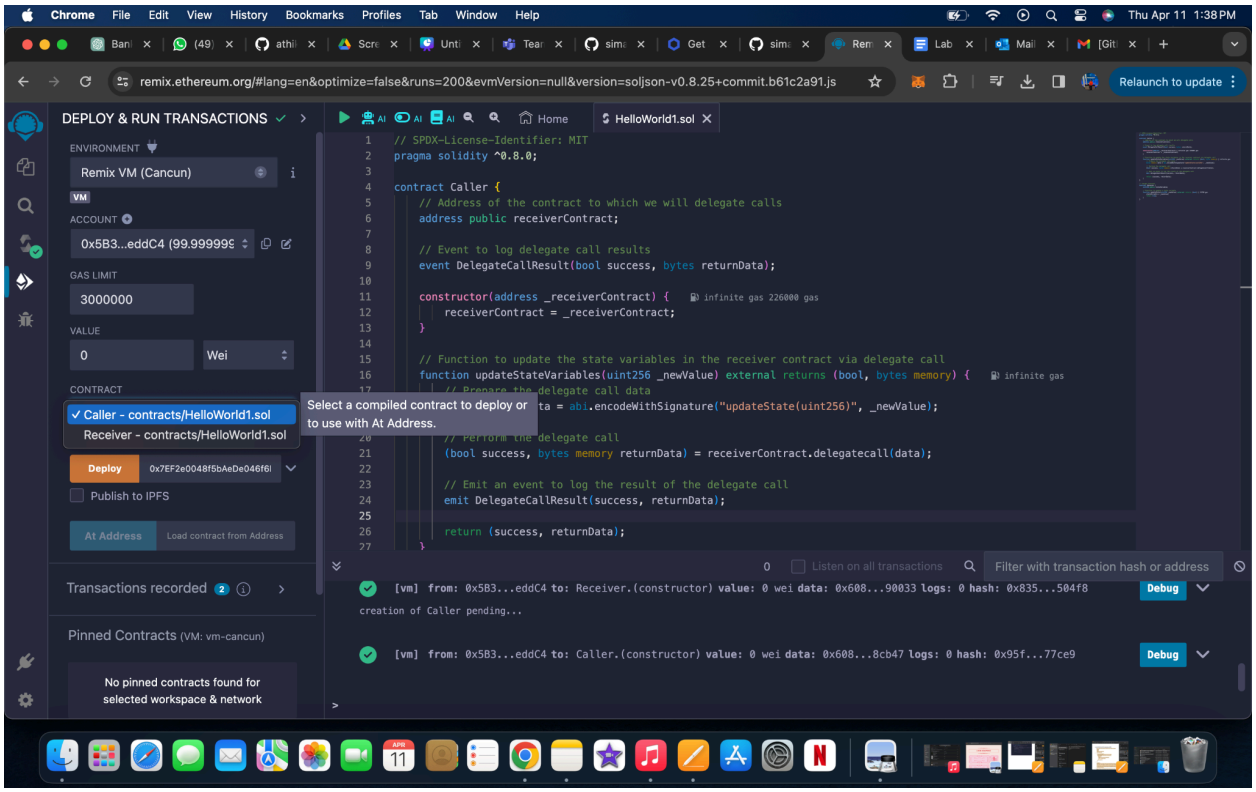
Cross-Chain Bridges:

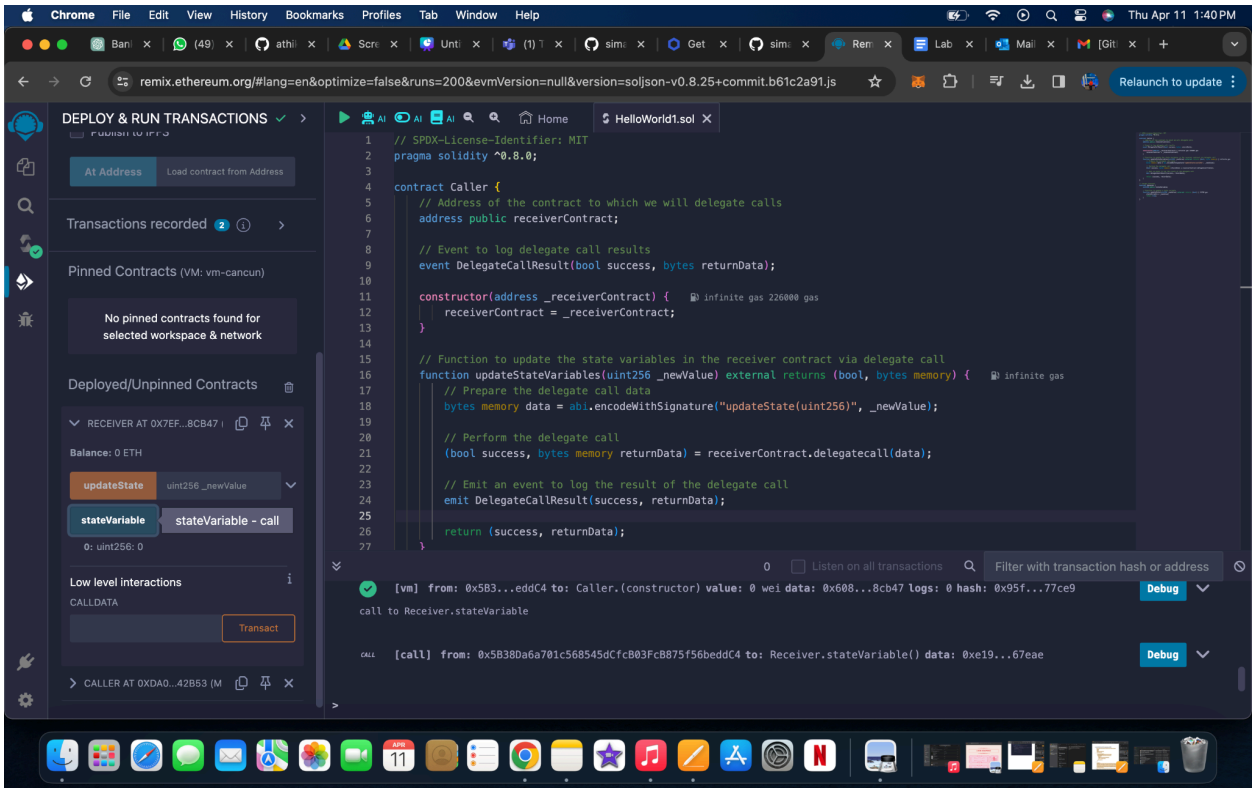
Advantages:

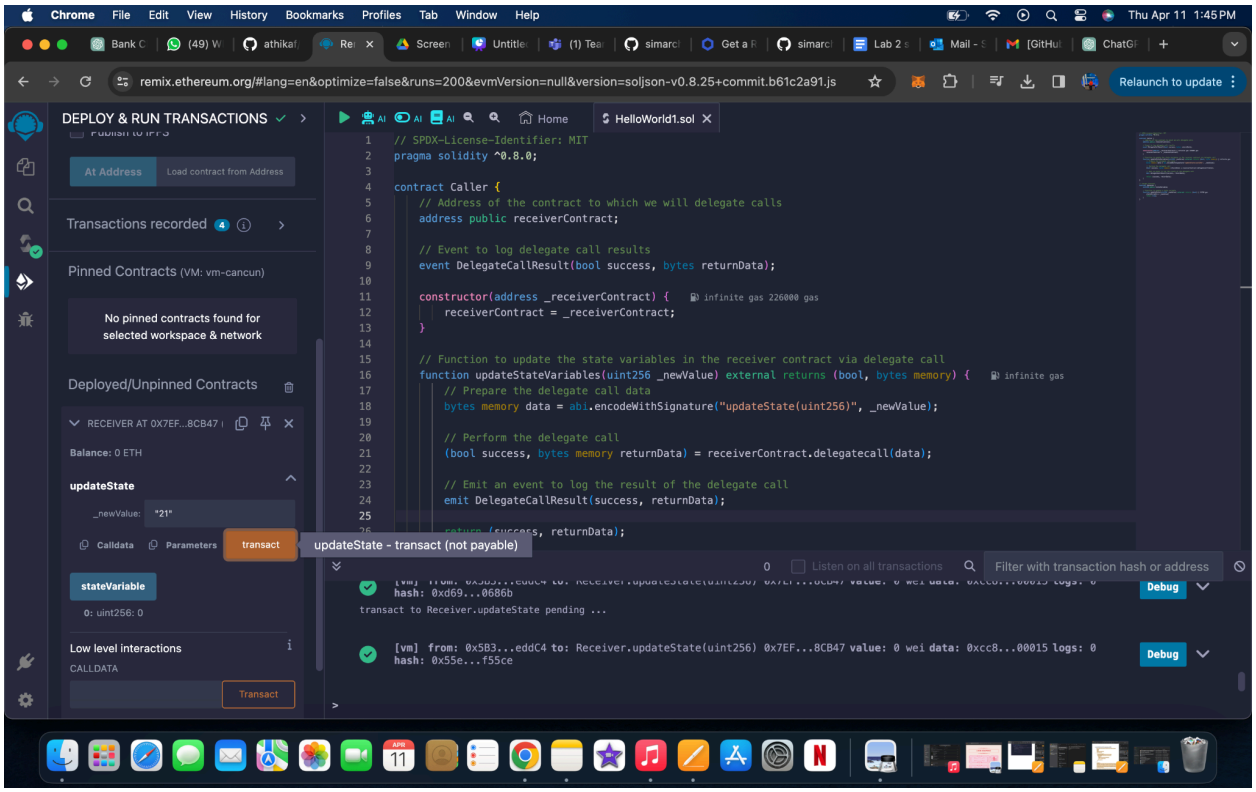
Wider Compatibility: Bridges connect blockchains with different architectures, enabling broader interoperability.
User-Friendly: Designed to be accessible, bridges shield users from technical complexities.
Liquidity Pools: Some bridges utilize liquidity pools to address liquidity issues, ensuring sufficient assets for swaps.
Disadvantages:

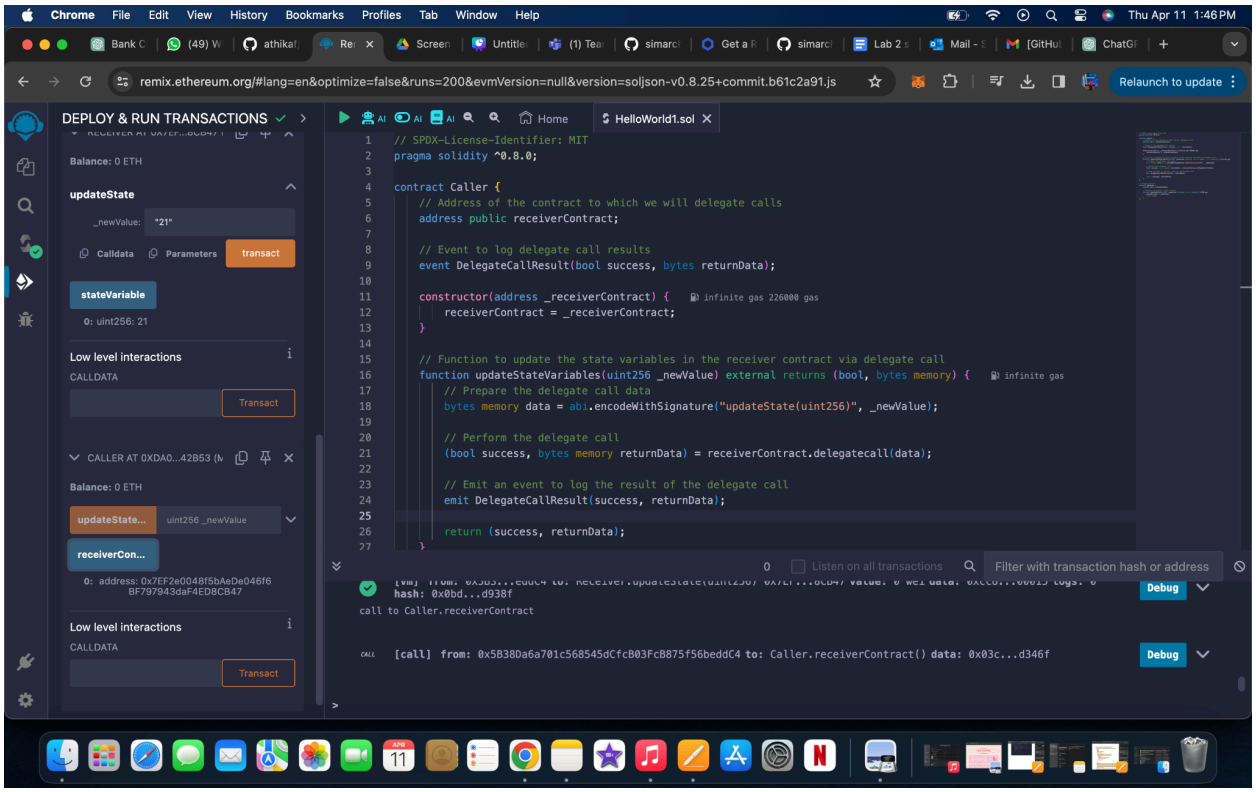
Centralization: Many bridges rely on validators or nodes, posing centralization risks if compromised.
Potential for Failure: Technical issues or exploits can lead to asset loss, requiring user trust in the bridge's security.
Complexity in Deployment: Building and maintaining bridges involves complexity and cost, requiring careful management to avoid vulnerabilities.
In conclusion, while atomic swaps offer decentralization and trustlessness, they may pose challenges in execution and compatibility. Cross-chain bridges provide wider compatibility and ease of use but introduce centralization risks and deployment complexities. Choosing between them depends on specific use cases and user preferences.

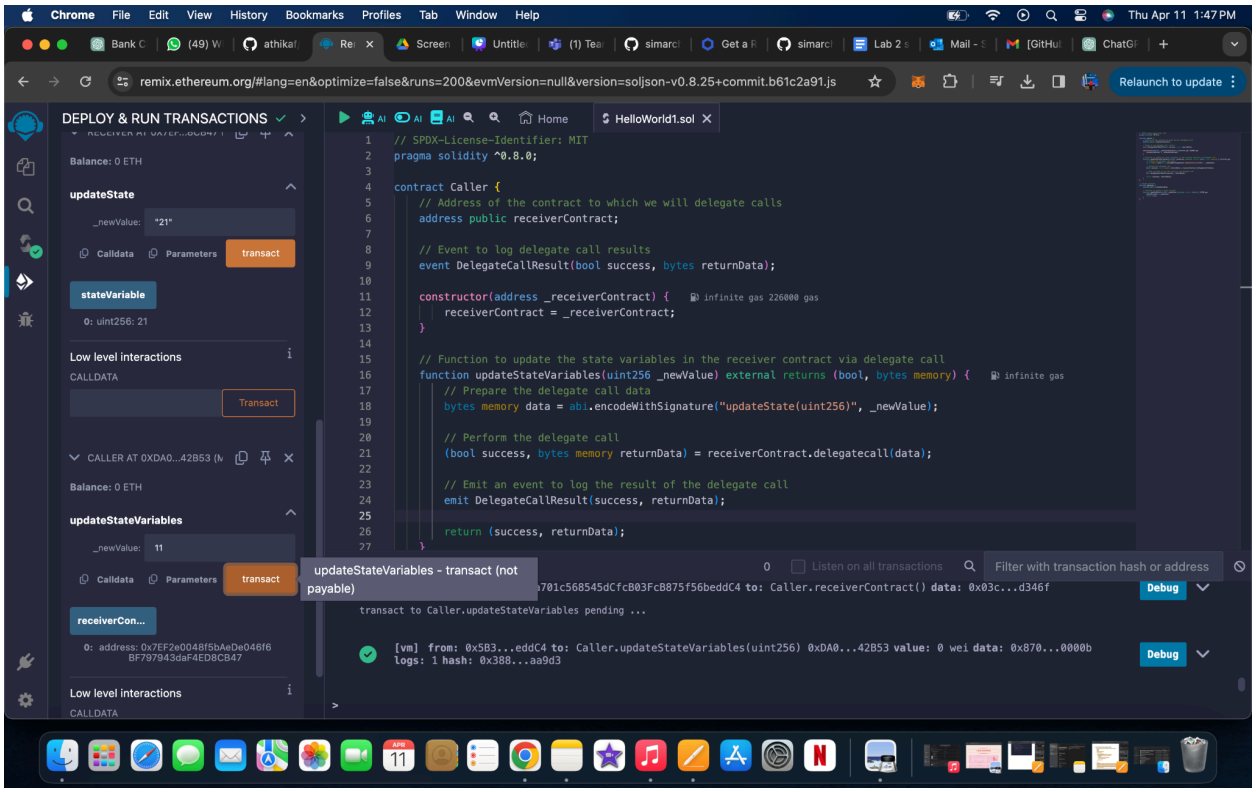



DEPLOY & RUN TRANSACTIONS
ENVIRONMENT
Remix VM (Cancun)
VM
ACCOUNT
0x5B3...eddC4 (99.99999
GAS LIMIT
3000000
VALUE
0
Wei
CONTRACT
Caller - contracts/HelloWorld1.sol
Receiver - contracts/HelloWorld1.sol
Select a compiled contract to deploy or to use with At Address.
Deploy
0x7EF2e0048f5bAeDe046f6
Publish to IPFS
At Address
Load contract from Address
Transactions recorded 2
Pinned Contracts (VM: vm-cancun)
No pinned contracts found for selected workspace & network
HelloWorld1.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract Caller {
// Address of the contract to which we will delegate calls
address public receiverContract;
// Event to log delegate call results
event DelegateCallResult(bool success, bytes returnData);
constructor(address _receiverContract) {
receiverContract = _receiverContract;
}
// Function to update the state variables in the receiver contract via delegate call
function updateStateVariables(uint256 _newValue) external returns (bool, bytes memory) {
// Perform the delegate call
(bool success, bytes memory returnData) = receiverContract.delegatecall(data);
// Emit an event to log the result of the delegate call
emit DelegateCallResult(success, returnData);
return (success, returnData);
}
Filter with transaction hash or address
[vm] from: 0x5B3...eddC4 to: Receiver.(constructor) value: 0 wei data: 0x608...90033 logs: 0 hash: 0x835...504f8
Debug
creation of Caller pending...
[vm] from: 0x5B3...eddC4 to: Caller.(constructor) value: 0 wei data: 0x608...8cb47 logs: 0 hash: 0x95f...77ce9
Debug


DEPLOY & RUN TRANSACTIONS
At Address
Load contract from Address
Transactions recorded 2
Pinned Contracts (VM: vm-cancun)
No pinned contracts found for selected workspace & network
Deployed/Unpinned Contracts
RECEIVER AT 0X7EF...8CB47
Balance: 0 ETH
updateState
uint256 _newValue
stateVariable
stateVariable - call
0: uint256: 0
Low level interactions
CALLDATA
Transact
CALLER AT 0XDA0...42B53 (M
HelloWorld1.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract Caller {
// Address of the contract to which we will delegate calls
address public receiverContract;
// Event to log delegate call results
event DelegateCallResult(bool success, bytes returnData);
constructor(address _receiverContract) {
receiverContract = _receiverContract;
}
// Function to update the state variables in the receiver contract via delegate call
function updateStateVariables(uint256 _newValue) external returns (bool, bytes memory) {
// Prepare the delegate call data
bytes memory data = abi.encodeWithSignature("updateState(uint256)", _newValue);
// Perform the delegate call
(bool success, bytes memory returnData) = receiverContract.delegatecall(data);
// Emit an event to log the result of the delegate call
emit DelegateCallResult(success, returnData);
return (success, returnData);
}
Filter with transaction hash or address
[vm] from: 0x5B3...eddC4 to: Caller.(constructor) value: 0 wei data: 0x608...8cb47 logs: 0 hash: 0x95f...77ce9
Debug
call to Receiver.stateVariable
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Receiver.stateVariable() data: 0xe19...67eae
Debug


DEPLOY & RUN TRANSACTIONS
At Address
Load contract from Address
Transactions recorded 4
Pinned Contracts (VM: vm-cancun)
No pinned contracts found for selected workspace & network
Deployed/Unpinned Contracts
RECEIVER AT 0X7EF...8CB47
Balance: 0 ETH
updateState
_newValue: "21"
Calldata
Parameters
transact
updateState - transact (not payable)
stateVariable
0: uint256: 0
Low level interactions
CALLDATA
Transact
Home
HelloWorld1.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract Caller {
// Address of the contract to which we will delegate calls
address public receiverContract;
// Event to log delegate call results
event DelegateCallResult(bool success, bytes returnData);
constructor(address _receiverContract) {
receiverContract = _receiverContract;
}
// Function to update the state variables in the receiver contract via delegate call
function updateStateVariables(uint256 _newValue) external returns (bool, bytes memory) {
// Prepare the delegate call data
bytes memory data = abi.encodeWithSignature("updateState(uint256)", _newValue);
// Perform the delegate call
(bool success, bytes memory returnData) = receiverContract.delegatecall(data);
// Emit an event to log the result of the delegate call
emit DelegateCallResult(success, returnData);
Listen on all transactions
Filter with transaction hash or address
hash: 0xd69...0686b
transact to Receiver.updateState pending ...
[vm] from: 0x5B3...eddC4 to: Receiver.updateState(uint256) 0x7EF...8CB47 value: 0 wei data: 0xcc8...00015 logs: 0
hash: 0x55e...f55ce
Debug
Debug


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Thu Apr 11 1:46 PM
remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.25+commit.b61c2a91.js
Relaunch to update
DEPLOY & RUN TRANSACTIONS
Balance: 0 ETH
updateState
_newValue: "21"
Calldata
Parameters
transact
stateVariable
0: uint256: 21
Low level interactions
CALLDATA
Transact
CALLER AT 0XDA0...42B53 (M
Balance: 0 ETH
updateState...
uint256 _newValue
receiverCon...
0: address: 0x7EF2e0048f5bAeDe046f6BF797943daF4ED8CB47
Low level interactions
CALLDATA
Transact
Home
HelloWorld1.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract Caller {
// Address of the contract to which we will delegate calls
address public receiverContract;
// Event to log delegate call results
event DelegateCallResult(bool success, bytes returnData);
constructor(address _receiverContract) { infinite gas 226000 gas
receiverContract = _receiverContract;
}
// Function to update the state variables in the receiver contract via delegate call
function updateStateVariables(uint256 _newValue) external returns (bool, bytes memory) { infinite gas
// Prepare the delegate call data
bytes memory data = abi.encodeWithSignature("updateState(uint256)", _newValue);
// Perform the delegate call
(bool success, bytes memory returnData) = receiverContract.delegatecall(data);
// Emit an event to log the result of the delegate call
emit DelegateCallResult(success, returnData);
return (success, returnData);
}
0
Listen on all transactions
Filter with transaction hash or address
hash: 0x0bd...d938f
Debug
call to Caller.receiverContract
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Caller.receiverContract() data: 0x03c...d346f
Debug


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Thu Apr 11 1:47 PM
remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.25+commit.b61c2a91.js
Relaunch to update
DEPLOY & RUN TRANSACTIONS
Balance: 0 ETH
updateState
_newValue: "21"
Calldata
Parameters
transact
stateVariable
0: uint256: 21
Low level interactions
CALLDATA
Transact
CALLER AT 0XDA0...42B53 (M
Balance: 0 ETH
updateStateVariables
_newValue: 11
Calldata
Parameters
transact
updateStateVariables - transact (not payable)
receiverCon...
0: address: 0x7EF2e0048f5bAeDe046f6BF797943daF4ED8CB47
Low level interactions
CALLDATA
Home
HelloWorld1.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract Caller {
// Address of the contract to which we will delegate calls
address public receiverContract;
// Event to log delegate call results
event DelegateCallResult(bool success, bytes returnData);
constructor(address _receiverContract) { infinite gas 226000 gas
receiverContract = _receiverContract;
}
// Function to update the state variables in the receiver contract via delegate call
function updateStateVariables(uint256 _newValue) external returns (bool, bytes memory) { infinite gas
// Prepare the delegate call data
bytes memory data = abi.encodeWithSignature("updateState(uint256)", _newValue);
// Perform the delegate call
(bool success, bytes memory returnData) = receiverContract.delegatecall(data);
// Emit an event to log the result of the delegate call
emit DelegateCallResult(success, returnData);
return (success, returnData);
}
0
Listen on all transactions
Filter with transaction hash or address
701c568545dCfcB03FcB875f56beddC4 to: Caller.receiverContract() data: 0x03c...d346f
Debug
transact to Caller.updateStateVariables pending ...
[vm] from: 0x5B3...eddC4 to: Caller.updateStateVariables(uint256) 0xDA0...42B53 value: 0 wei data: 0x870...0000b
logs: 1 hash: 0x388...aa9d3
Debug