SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		**Assignment Type:** Lab	**Academic Year:** 2025-26
Course Coordinator Name		Dr. Jagjit Singh Dhatterwal	
Instructor(s) Name		Dr. Jagjit Singh Dhatterwal	
Course Code	23CS201PE401	**Course Title**	Blockchain Engineering
Year/Sem	III/II	**Regulation**	R25
Date and Day of Assignment	12-01-2026	**Time(s)**	9:00AM to 11:00AM
Duration	2 Hours	**Applicable to Batches**	(23CSBTB19, 23CSBTB20 23CSBTB21, 23CSBTB22 23CSBTB23, 23CSBTB24 23CSBTB25, 23CSBTB26)
Assignment Number: 02/12			

Q. No.	Question	Expected Time to complete
1	**Title** To design, deploy, and interact with a Simple Storage smart contract using the Solidity programming language in the Remix Ethereum IDE, in order to demonstrate fundamental blockchain concepts such as state variables, smart contract functions, and blockchain transactions. **Tools / Software Required** • Web Browser (Chrome / Firefox) • **Remix Ethereum IDE** (online) • Solidity Compiler (built-in with Remix) • JavaScript VM (Remix default environment) **Theory (Brief)** A **smart contract** is a self-executing program stored on the blockchain. The **Simple Storage contract** is a beginner-level Solidity contract that allows users to: • Store a value on the blockchain • Retrieve the stored value Remix IDE is a web-based development environment used to write, compile, deploy, and test Solidity smart contracts.	

Requirements

- Create a Solidity smart contract named `SimpleStorage`
- Declare a state variable to store data
- Implement functions to:
  - Store a value
  - Retrieve the stored value
- Deploy the contract using Remix
- Test the contract with multiple inputs
- Add comments explaining the logic
- Capture screenshots of:
  - Contract compilation
  - Contract deployment
  - Function execution

Procedure / Steps

Step 1: Open Remix IDE

1. Open a web browser
2. Go to: https://remix.ethereum.org
3. Create a new file named:
4. `SimpleStorage.sol`

Step 2: Write the Solidity Code

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

// Simple Storage Smart Contract
contract SimpleStorage {

    // State variable to store a number
    uint256 private storedValue;

    // Function to store a value on the
blockchain
    function set(uint256 _value) public {
        storedValue = _value;
    }

    // Function to retrieve the stored value
    function get() public view returns
(uint256) {
        return storedValue;
    }
}
```

Code Explanation

- `pragma solidity ^0.8.0;`
  Specifies the Solidity compiler version.
- `uint256 private storedValue;`
  Declares a private state variable stored on the blockchain.
- `set(uint256 _value)`
  Stores a value by creating a transaction.
- `get()`
  Reads the stored value without modifying blockchain data.

Step 3: Compile the Contract

1. Click on **Solidity Compiler** tab
2. Select compiler version `0.8.x`
3. Click **Compile SimpleStorage.sol**
4. Ensure there are **no errors**

Step 4: Deploy the Contract

1. Go to **Deploy & Run Transactions** tab
2. Select **JavaScript VM (London)** as environment
3. Click **Deploy**
4. Contract instance appears under **Deployed Contracts**

Code Screenshot GUI:

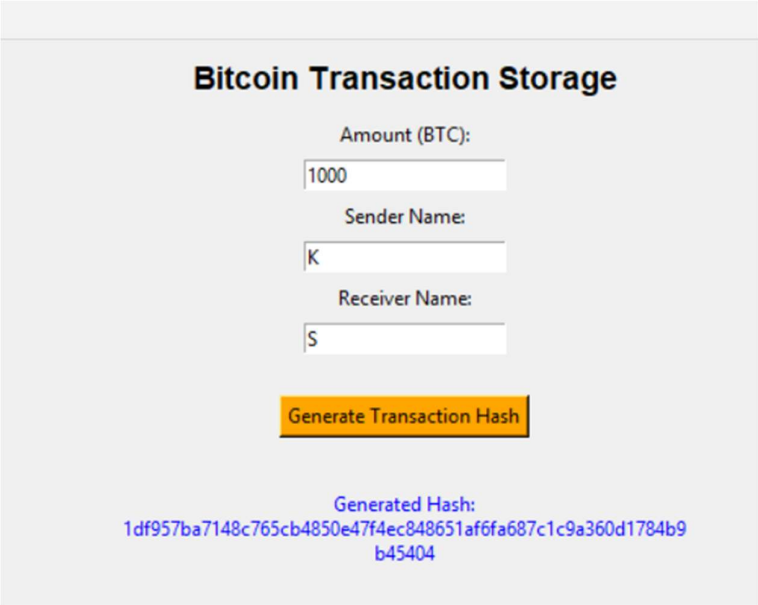
PYTHON_

```python
import tkinter as tk
from tkinter import messagebox
import hashlib

def generate_hash():
    # Get values from the GUI input boxes
    amount = entry_amount.get()
    sender = entry_sender.get()
    receiver = entry_receiver.get()

    if not amount or not sender or not receiver:
        messagebox.showwarning("Input Error", "Please fill all fields")
        return

    # Create the Hashing Value (The Encryption Check)
    raw_data = f"{amount}{sender}{receiver}"
    tx_hash = hashlib.sha256(raw_data.encode()).hexdigest()

    # Display the result in the GUI
    label_hash_result.config(text=f"Generated Hash:\n{tx_hash}")

    # Instructions for the user
    print(f"--- Copy this to Remix ---")
    print(f"Amount: {amount}")
    print(f"Hash: {tx_hash}")
```

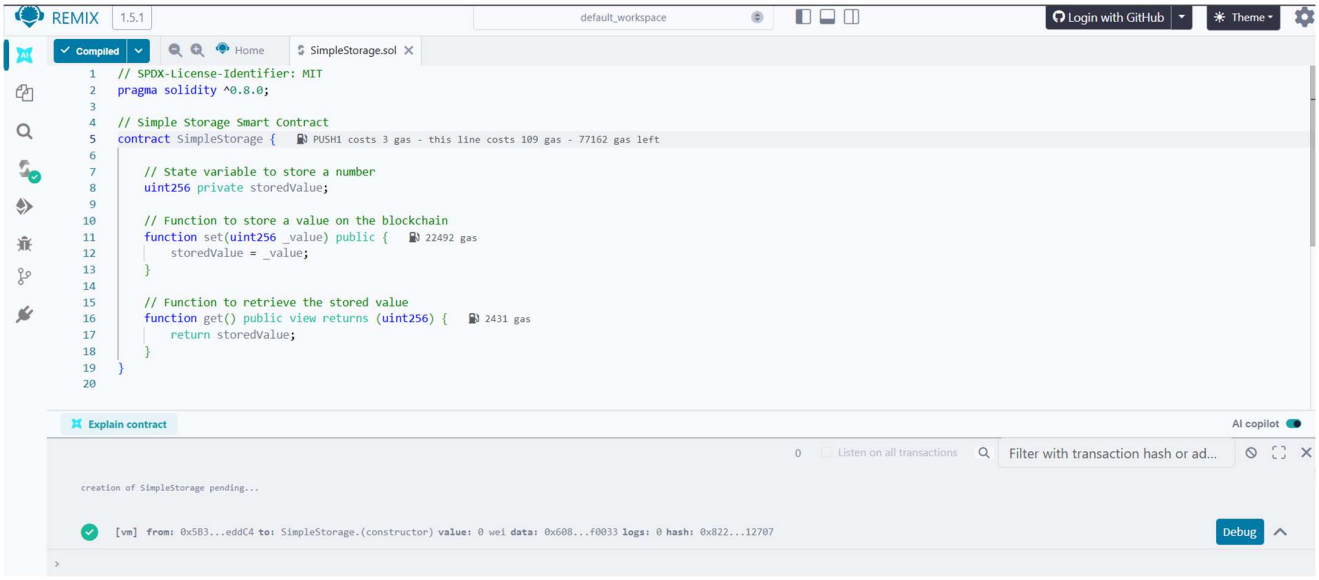
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL HISTORY TASK MONITOR

```
--- Copy this to Remix ---
Amount: 1000
Hash: 1df957ba7148c765cb4850e47f4ec848651af6fa687c1c9a360d1784b9b45404
--- Copy this to Remix ---
Amount: 1000
Hash: 1df957ba7148c765cb4850e47f4ec848651af6fa687c1c9a360d1784b9b45404
--- Copy this to Remix ---
Amount: 1000
Hash: 1df957ba7148c765cb4850e47f4ec848651af6fa687c1c9a360d1784b9b45404
```

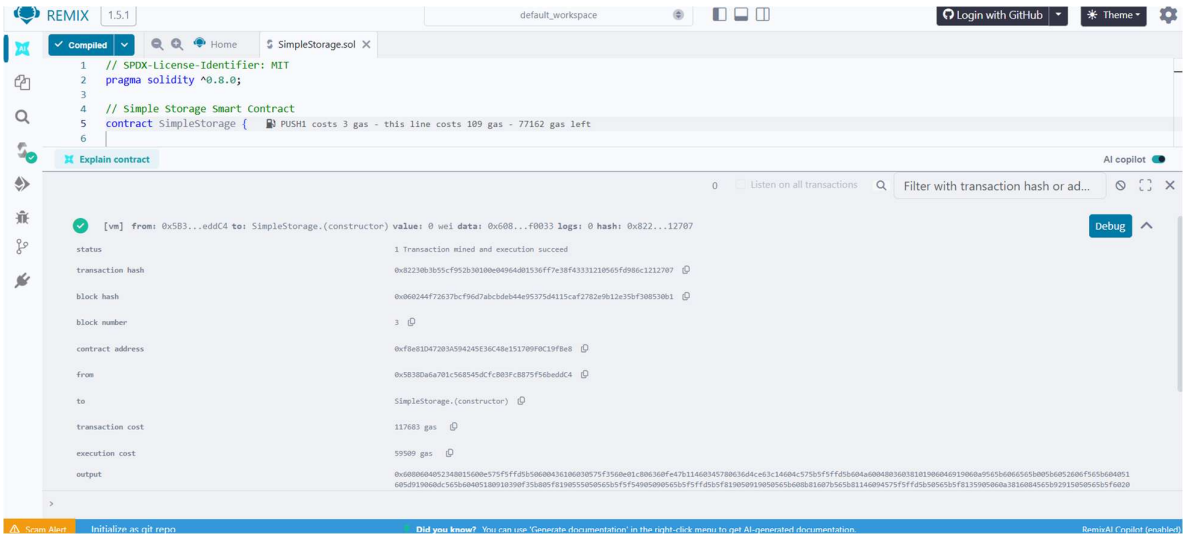
Output :

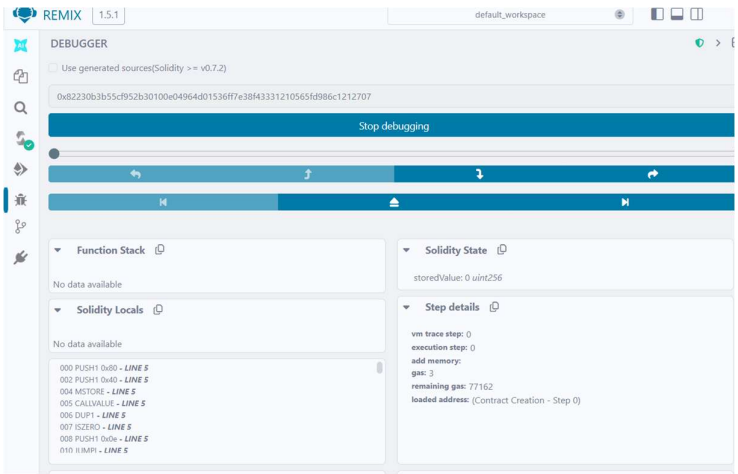


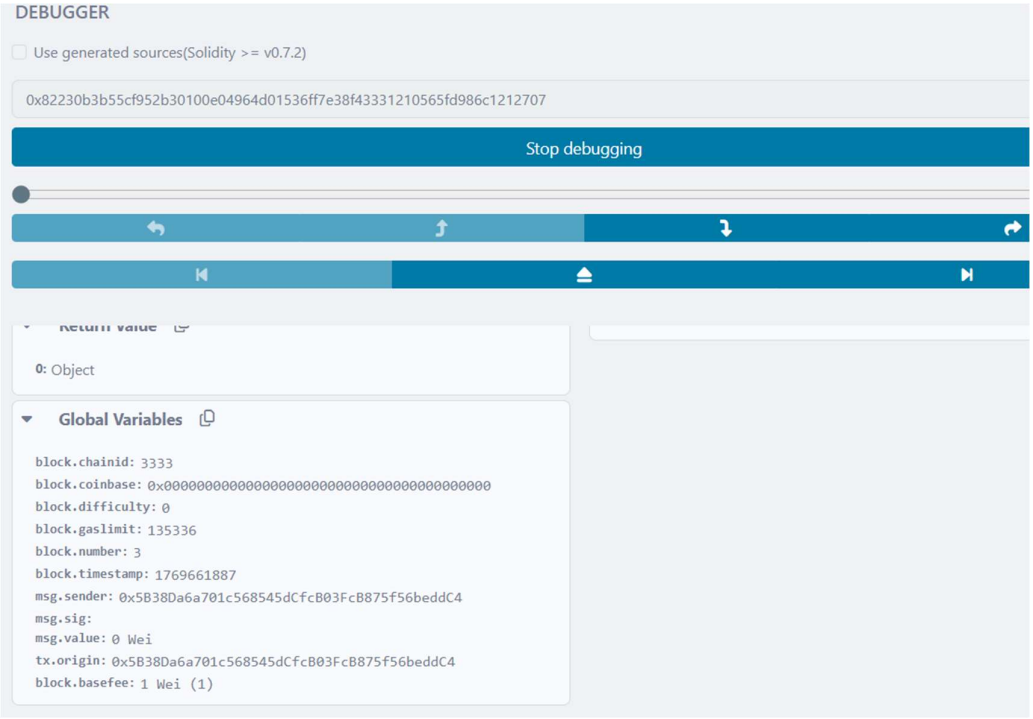
Ethernium Code :



Output :






DEBUGGER
Use generated sources(Solidity >= v0.7.2)
0x82230b3b55cf952b30100e04964d01536ff7e38f43331210565fd986c1212707
Stop debugging
0: Object
Global Variables
block.chainid: 3333
block.coinbase: 0x0000000000000000000000000000000000000000
block.difficulty: 0
block.gaslimit: 135336
block.number: 3
block.timestamp: 1769661887
msg.sender: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
msg.sig:
msg.value: 0 Wei
tx.origin: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
block.basefee: 1 Wei (1)