

NAAN MUDHALVAN – BLOCKCHAIN

ASSIGNMENT-1

TEAM ID: NM2023TMID09517

TEAM LEAD: PREETHI S

TEAM MEMBER 1: HARISH UP

TEAM MEMBER 2: RAJESHWARI S

TEAM MEMBER 3: DEEPA M

QUESTION:

Display a zone name as an output. Deploy the code by using remix platform.

Solution:

Source code:

```
// SPDX-License-Identifier: MIT
```

```
pragma solidity ^0.8.0;
```

```
contract MyContract {
```

```
    function getZoneName() public pure returns (string memory) {
```

```
        return "Zone - 4";
```

```
    }
```

```
}
```

Code explanation:

1. **// SPDX-License-Identifier: MIT:** This is a comment at the beginning of the contract, indicating the license under which the code is released. In this case, it specifies the MIT license, which is a permissive open-source license.

2. **pragma solidity ^0.8.0;** This line specifies the Solidity compiler version to be used for compiling the contract. The caret (^) symbol (^0.8.0) indicates that any compiler version from 0.8.0 up to, but not including, 0.9.0 is acceptable. This ensures that the code is compiled with a compatible compiler version.

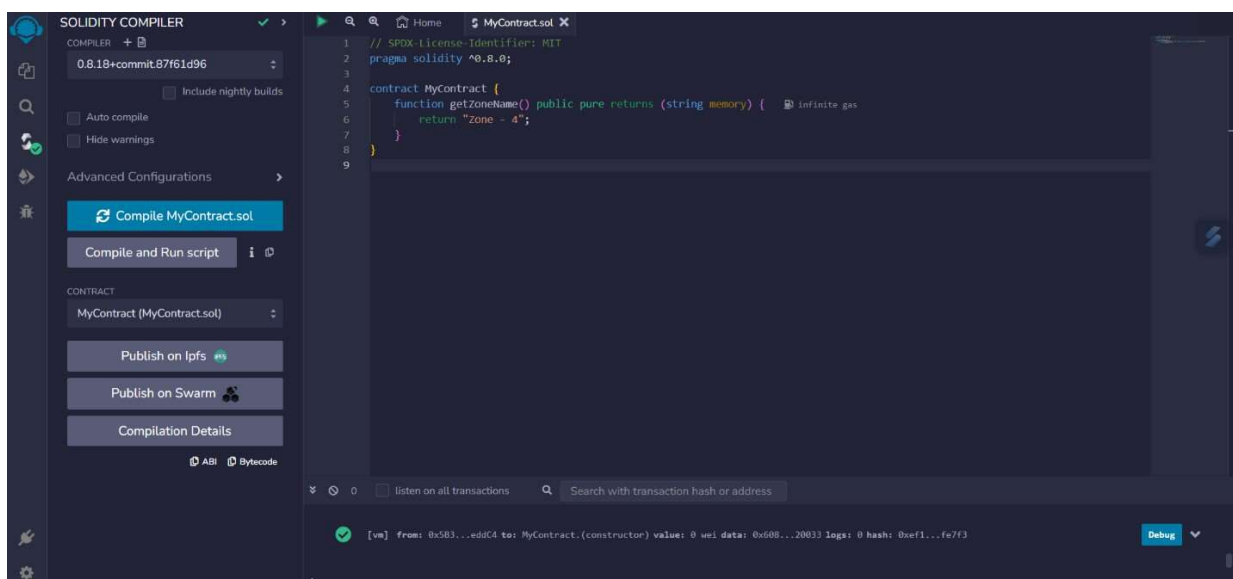
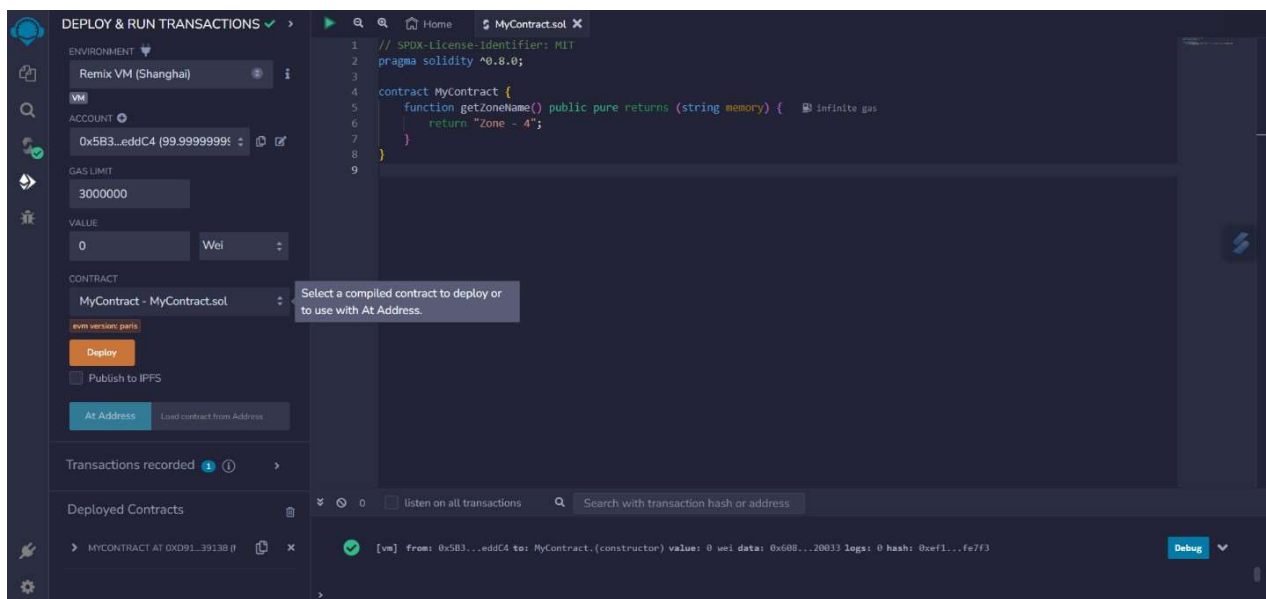
3. **contract MyContract { ... }:** This defines a smart contract named `MyContract`. In Solidity, a contract is a fundamental building block of Ethereum-based applications.

4. **function getZoneName() public pure returns (string memory) { ... }:** This is a function within the `MyContract` contract. Let's break down the function:

- **function getZoneName():** This declares a function named `getZoneName`. It's a publicly accessible function, meaning it can be called from outside the contract.
- **public:** This is a visibility specifier, indicating that the function is accessible from outside the contract. It can be called by anyone who interacts with the contract.
- **pure:** This is a function state mutability specifier. A `pure` function indicates that it doesn't modify the contract's state. It's used for functions that perform computations but don't change any data in the contract. In this case, the function simply returns a value and doesn't modify any contract state.

- **returns (string memory):** This specifies the return type of the function. The function returns a dynamic string stored in memory. Solidity functions can return different data types, and in this case, it's returning a string.
- **{ return "Zone - 4"; }:** This is the body of the function. It simply returns the string "Zone - 4."

Screenshots:



DEPLOY & RUN TRANSACTIONS

GAS LIMIT

3000000

VALUE

0

Wei

CONTRACT

MyContract - MyContract.sol

even version pairs

Deploy

Publish to IPFS

At Address

Load contract from Address

Transactions recorded

Deployed Contracts

MYCONTRACT AT 0XD91...39138

Balance: 0 ETH

getZoneName

string: Zone - 4

Low level interactions

CALLDATA

Transact

MyContract.sol

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract MyContract {
5     function getZoneName() public pure returns (string memory) {
6         return "Zone - 4";
7     }
8 }
```

0

listen on all transactions

Search with transaction hash or address

ethers.js

remix

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

✓

[vm] from: 0x5B3...eddC4 to: MyContract.(constructor) value: 0 wei data: 0x688...20833 logs: 0 hash: 0xef1...fe7f3

call to MyContract.getZoneName

Debug

ox:

[call] from: 0x5B38Da6a701c56854dCfcb03FcB875F56beddC4 to: MyContract.getZoneName() data: 0x235...ed78f

Debug

from

0x5B38Da6a701c56854dCfcb03FcB875F56beddC4

to

MyContract.getZoneName() 0xd9145cCE520386f254917e483e844e943f39138

execution cost

715 gas (Cost only applies when called by a contract)

input

0x235...ed78f

decoded input

[]

decoded output

{ "0": "string: Zone - 4" }

logs

[]
