Name:- Anjali punsi
Class: D20B
Roll no:- 57
Experiment no:- 6

AIM: Create ERC 20 token using Remix IDE/ truffle and ganache**.**

Lab Objectives: To learn to Create ERC 20 token.

Lab Outcomes (LO): (LO4)

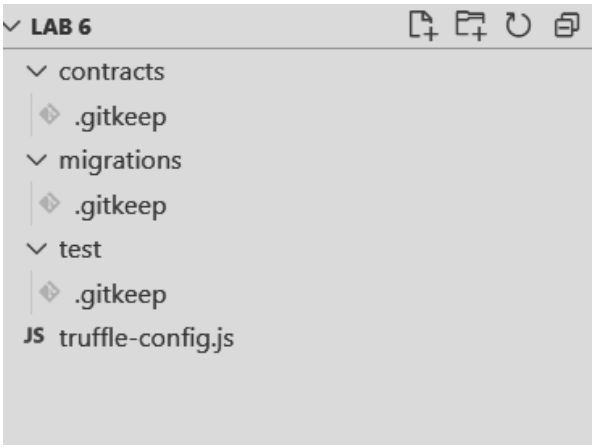
The task to be performed: Create ERC20 Token

Tools & Libraries used: Remix IDE and metamask/ truffle and ganache.

Steps:-
Step 1: Initialize your project

```
C:\Users\Anjali\Downloads\lab 6>truffle init

Starting init...
================

> Copying project files to C:\Users\Anjali\Downloads\lab 6

Init successful, sweet!

Try our scaffold commands to get started:
  $ truffle create contract YourContractName # scaffold a contract
  $ truffle create test YourTestName         # scaffold a test

http://trufflesuite.com/docs


C:\Users\Anjali\Downloads\lab 6>
```

Step 2: Once this operation is complete, you'll now have a project structure with the following items:



Step 3: Create a file named “MyToken.sol” in the contract directory.

Code :-

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.5.16;

contract MyToken {
    string public name;
    string public symbol;
    uint8 public decimals;
    uint256 public totalSupply;

    mapping(address => uint256) public balanceOf;
    mapping(address => mapping(address => uint256)) public allowance;

    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);

    constructor(string memory _name, string memory _symbol, uint8 _decimals, uint256
_initialSupply) public {
        name = _name;
        symbol = _symbol;
        decimals = _decimals;
        totalSupply = _initialSupply * 10 ** uint256(_decimals);
        balanceOf[msg.sender] = totalSupply;
```

```
    }

    function transfer(address _to, uint256 _value) public returns (bool success) {
        require(balanceOf[msg.sender] >= _value, "Insufficient balance");
        balanceOf[msg.sender] -= _value;
        balanceOf[_to] += _value;
        emit Transfer(msg.sender, _to, _value);
        return true;
    }

    function approve(address _spender, uint256 _value) public returns (bool success) {
        allowance[msg.sender][_spender] = _value;
        emit Approval(msg.sender, _spender, _value);
        return true;
    }

    function transferFrom(address _from, address _to, uint256 _value) public returns (bool
success) {
        require(_value <= balanceOf[_from], "Insufficient balance");
        require(_value <= allowance[_from][msg.sender], "Allowance exceeded");
        balanceOf[_from] -= _value;
        balanceOf[_to] += _value;
        allowance[_from][msg.sender] -= _value;
        emit Transfer(_from, _to, _value);
        return true;
    }
}
```

Step 4: Compile a Truffle project

```
Node v16.17.0
PS C:\Users\Anjali\Downloads\lab 6> npx truffle compile

Compiling your contracts...
===========================
> Compiling .\contracts\MyToken.sol
> Compilation warnings encountered:

    Warning: Visibility for constructor is ignored. If you want the contract to be non-deployable, m
  --> project:/contracts/MyToken.sol:16:5:
   |
16 |     constructor(string memory _name, string memory _symbol, uint8 _decimals, uint256 _initialSu
   |     ^ (Relevant source part starts here and spans across multiple lines).


> Artifacts written to C:\Users\Anjali\Downloads\lab 6\build\contracts
> Compiled successfully using:
   - solc: 0.8.21+commit.d9974bed.Emscripten.clang
PS C:\Users\Anjali\Downloads\lab 6>
```

Step 5: Create a migration to get the contract on the network. Create a file in the migrations folder named “2_Anjali_migrate_deploy.js”.

migrations > JS 2_Anjali_migrate_deploy.js > ...

Click here to ask Blackbox to help you code faster

```
1  const MyToken = artifacts.require("MyToken");
2
3  module.exports = function (deployer) {
4    // Deployment of the ERC20 token contract
5    deployer.deploy(MyToken, "MyToken", "MTK", 18, 1000000);
6  };
7
```

Code:-

```
const MyToken = artifacts.require("MyToken");

module.exports = function (deployer) {
  // Deployment of the ERC20 token contract
  deployer.deploy(MyToken, "MyToken", "MTK", 18, 1000000);
};
```

Step 6: The ERC20 Token Smart contract will be deployed

```
PS C:\Users\Anjali\Downloads\lab 6> npx truffle migrate --network development
```

```
PS C:\Users\Anjali\Downloads\lab 6> npx truffle migrate --network development

Compiling your contracts...
===========================
> Compiling .\contracts\MyToken.sol
> Compiling .\contracts\MyToken.sol
> Artifacts written to C:\Users\Anjali\Downloads\lab 6\build\contracts
> Compiled successfully using:
   - solc: 0.8.21+commit.d9974bed.Emscripten.clang
> Something went wrong while attempting to connect to the network at http://127.0.0.1:8545. Check your network configuration.
CONNECTION ERROR: Couldn't connect to node http://127.0.0.1:8545.
Truffle v5.11.5 (core: 5.11.5)
Node v16.17.0
```

Conclusion :

In conclusion, troubleshooting issues with connecting Truffle to an Ethereum network involves checking the network availability, configuration, and potential conflicts with other processes or software. By following the steps outlined above, you can resolve common connectivity issues and successfully deploy your contracts. If you continue to encounter problems, providing detailed information about your setup and any error messages will help in further troubleshooting.