Name : Pratik Tiruwa
Seat No : 31031523035

Practical 12

A. Create a DApp to calculate factorial of a number.

1. In a new terminal run truffle init

2. Create a new contract to calculate Factorial.

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

pragma solidity 0.8.19;

contract factorial {
    function fact(uint n) public pure returns (uint) {
        if (n == 0) {
            return 1;
        } else {
            uint result = 1;
            for (uint i = 1; i <= n; i++) {
                result *= i;
            }
            return result;
        }
    }
}
```

3.. Make a new folder frontend and create two files, index.html & app.js.index.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>DApp-2</title>
    </head>
    <body>
        <h1>Blockchain Factorial DApp</h1>
        Number:
```

```
        <input type="number" id="num" placeholder="Enter Number" />
        <br /><br />
        <h2>Calculate Factorial:</h2>
        <button onclick="facto()">Calculate</button>
        <h2>Result: <span id="result"></span></h2>
        <script
src="https://cdn.jsdelivr.net/npm/web3@latest/dist/web3.min.js"></script>
        <script src="app.js"></script>
    </body>
</html>
```

app.js (get contractABI & contractAddress after compilation and migration respectively)

Name : Pratik Tiruwa
Seat No : 31031523035

```
const contractAddress = ""; // Replace with your deployed contract address
const contractABI = []; // Use ABI from compiled contract

let web3;
let contract;

window.addEventListener("load", async () => {
    if (window.ethereum) {
        web3 = new Web3(window.ethereum);
        await window.ethereum.enable();
    } else {
        console.log("MetaMask not detected. Please install MetaMask.");
    }

    contract = new web3.eth.Contract(contractABI, contractAddress);
});

async function facto() {
    const num = document.getElementById("num").value;
    const accounts = await web3.eth.getAccounts();
    console.log(num);
    contract.methods
```

```
        .fact(num)
        .call({ from: accounts[0] })
        .then((result) => {
            console.log(result);
            document.getElementById("result").innerText = `${result}`;
        });
}
```

4. Create 1_deploy.js in migrations folder.

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

module.exports = async function (deployer) {
  await deployer.deploy(factorial);
  const instance = await factorial.deployed();
  console.log("Operations deployed at:", instance.address);
};
```

5. Run the DApp by npm start. Connect wallet and test.