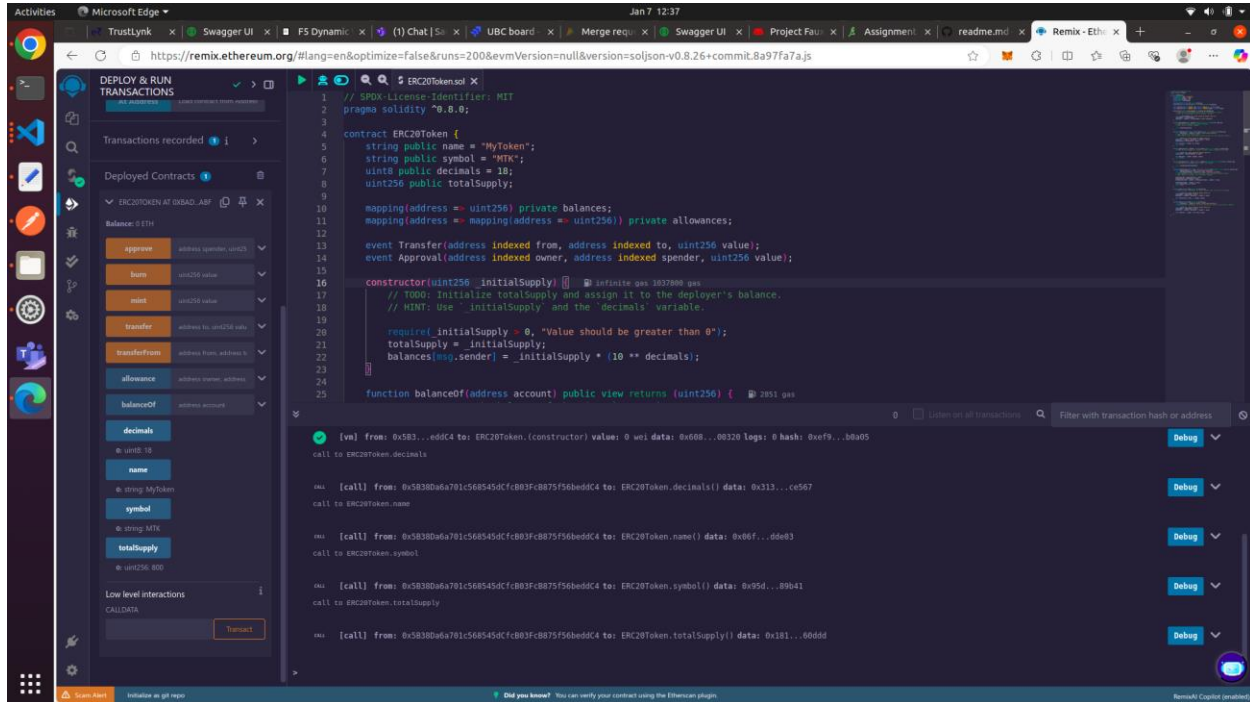
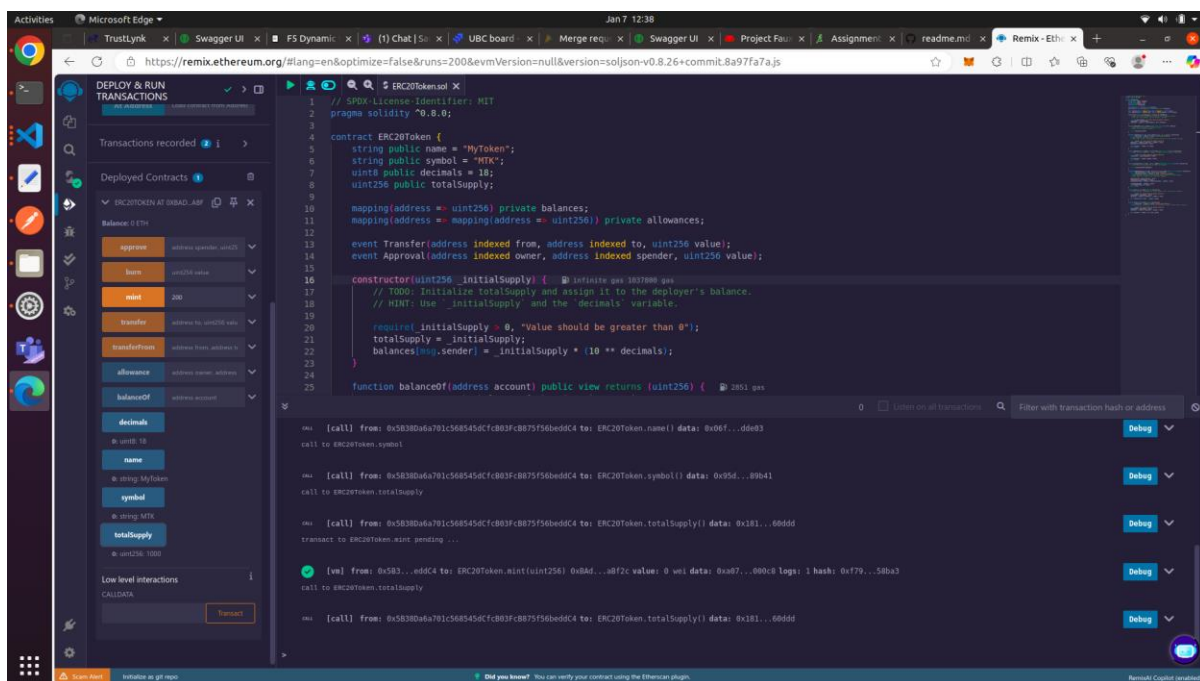


SNAPSHOTS (Student Name: Harsh Gupta (NPCI))

1. **Deploy Contract:** Contract deployed, and initial supply is 800 Ethers. Also, you can view the balance, totalSupply, name and symbol.



2. **Mint:** Minting 200 and the totalSupply is 1000 Now.



3. Burn: Burnt 100 and now the totalSupply is 900 Now.

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is open, displaying the 'ERC20Token' contract. The 'Burn' function is selected, and the 'Burnt' value is set to 100. The 'Total Supply' is shown as 900. The main editor displays the Solidity code for the ERC20Token contract. The transaction log at the bottom shows the following transactions:

- [call] from: 0x5830d6a701c568545dcf803fc8b75f56bedd4 to: ERC20Token.totalSupply() data: 0x181...60dd
- [call] from: 0x583...edd4 to: ERC20Token.mint(uint256) 0x8d...ab7c value: 0 wei data: 0xad7...0064 logs: 1 hash: 0xf79...58ba3
- [call] from: 0x5830d6a701c568545dcf803fc8b75f56bedd4 to: ERC20Token.totalSupply() data: 0x181...60dd
- [call] from: 0x583...edd4 to: ERC20Token.burn(uint256) 0x8d...ab7c value: 0 wei data: 0x429...0064 logs: 1 hash: 0x2ca...c7647
- [call] from: 0x5830d6a701c568545dcf803fc8b75f56bedd4 to: ERC20Token.totalSupply() data: 0x181...60dd

4. Transfer: Transfer when balance is sufficient.

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is open, displaying the 'ERC20Token' contract. The 'Transfer' function is selected, and the 'To' address is set to '0x5830d6a701c568545dcf803fc8b75f56bedd4'. The 'Value' is set to 100. The 'Total Supply' is shown as 900. The main editor displays the Solidity code for the ERC20Token contract. The transaction log at the bottom shows the following transactions:

- [call] from: 0x583...35cb2 to: ERC20Token.mint(uint256) 0x8d...ab7c value: 0 wei data: 0xad7...0064 logs: 1 hash: 0x3ba...06106
- [call] from: 0x583...35cb2 to: ERC20Token.transfer(address,uint256) 0x8d...ab7c value: 0 wei data: 0xa90...0012c logs: 0 hash: 0x0b6...fe706
- [call] from: 0x583...35cb2 to: ERC20Token.transfer(address,uint256) 0x8d...ab7c value: 0 wei data: 0xa90...0012c logs: 1 hash: 0xb2f...850ea

5. Allowance and Approve: Adding allowance and approve.

The screenshot shows the Remix IDE interface with the ERC20Token contract code open. The code includes functions for minting, burning, and transferring tokens, as well as allowance and approval logic. The transaction logs on the right show the execution of the `approve` function, which successfully updates the allowance for the specified address.

```
69 allowances[from[msg.sender]] = allowances[from[msg.sender]] + value;
70
71 allowances[from[msg.sender]] -= value;
72 emit Transfer(from, to, value);
73 return true;
74
75 function mint(uint256 value) public {
76     // TODO: Increase 'totalSupply' by 'value' and add 'value' to the caller's balance.
77
78     require(value > 0, "Value should be greater than 0");
79     totalSupply = totalSupply + value;
80     balances[msg.sender] = balances[msg.sender] + value;
81     emit Transfer(address(0), msg.sender, value);
82 }
83
84 function burn(uint256 value) public {
85     // TODO: Decrease 'totalSupply' by 'value' and subtract 'value' from the caller's balance.
86
87     require(balances[msg.sender] >= value, "Insufficient balance");
88     // TODO: Decrease 'totalSupply' by 'value' and subtract 'value' from the caller's balance.
89     require(value > 0, "Value should be greater than 0");
90     totalSupply = totalSupply - value;
91     balances[msg.sender] = balances[msg.sender] - value;
92     emit Transfer(msg.sender, address(0), value);
93 }
```

Transaction logs:

- [vm] from: 0xA88...35cb2 to: ERC20Token.transfer(address,uint256) 0xA88...a87c value: 0 wei data: 0xa99...8012c logs: 0 hash: 0x96b...fef86
- transaction to ERC20Token.transfer failed: Error occurred: revert.
- revert
- The transaction has been reverted to the initial state.
- Reason provided by the contract: "Insufficient balance".
- If the transaction failed for not having enough gas, try increasing the gas limit gently.
- transaction to ERC20Token.transfer pending ...
- [vm] from: 0xA88...35cb2 to: ERC20Token.transfer(address,uint256) 0xA88...a87c value: 0 wei data: 0xa99...00032 logs: 1 hash: 0xb2f...85aea
- transaction to ERC20Token.approve pending ...
- [vm] from: 0x583...edd4 to: ERC20Token.approve(address,uint256) 0xA88...a87c value: 0 wei data: 0x995...000c8 logs: 1 hash: 0xe92...5e5f6

6. Balance: Fetching balance for an account.

The screenshot shows the Remix IDE interface with the ERC20Token contract code open. The code includes functions for minting, burning, and transferring tokens, as well as allowance and approval logic. The transaction logs on the right show the execution of the `balanceOf` function, which successfully returns the balance for the specified address.

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract ERC20Token {
5     string public name = "MyToken";
6     string public symbol = "MTK";
7     uint8 public decimals = 18;
8     uint256 public totalSupply;
9
10     mapping(address => uint256) private balances;
11     mapping(address => mapping(address => uint256)) private allowances;
12
13     event Transfer(address indexed from, address indexed to, uint256 value);
14     event Approval(address indexed owner, address indexed spender, uint256 value);
15
16     constructor(uint256 _initialSupply) {
17         // TODO: Initialize totalSupply and assign it to the deployer's balance.
18         // HINT: Use '_initialSupply' and the 'decimals' variable.
19
20         require(_initialSupply > 0, "Value should be greater than 0");
21         totalSupply = _initialSupply;
22         balances[msg.sender] = _initialSupply * (10 ** decimals);
23     }
24
25     function balanceOf(address account) public view returns (uint256) {
26         // TODO: Return the balance of the account.
27     }
28 }
```

Transaction logs:

- [vm] from: 0x383...edd4 to: ERC20Token.mint(uint256) 0xA88...a87c value: 0 wei data: 0xad7...000c8 logs: 1 hash: 0xf79...58ba3
- call to ERC20Token.totalSupply
- [call] from: 0x58380d6a701c568545cf803fc8073f56bed0c4 to: ERC20Token.totalSupply() data: 0x181...60dd9
- transaction to ERC20Token.burn pending ...
- [vm] from: 0x383...edd4 to: ERC20Token.burn(uint256) 0xA88...a87c value: 0 wei data: 0x429...00064 logs: 1 hash: 0x2ca...c6047
- call to ERC20Token.totalSupply
- [call] from: 0x58380d6a701c568545cf803fc8073f56bed0c4 to: ERC20Token.totalSupply() data: 0x181...60dd9
- call to ERC20Token.balanceOf
- [call] from: 0x58380d6a701c568545cf803fc8073f56bed0c4 to: ERC20Token.balanceOf(address) data: 0x7ba...edd4

ERROR Cases:

1. Transfer: Insufficient balance

The screenshot displays the Remix IDE interface with the following components:

- Left Panel (Deploy & Run Transactions):** Shows the deployed contract 'ERC20Token' at address '0x5830d6a701c568545dcf8b3f8b75f56bed0c4'. The 'transfer' function is selected, with parameters: 'to: 0x5830d6a701c568545dcf8b3f8b75f56bed0c4' and 'value: 8000'.
- Center Panel (Solidity Code):** Displays the contract code for 'ERC20Token'.
- Right Panel (Transaction History):** Shows a list of transactions. The most recent transaction is a failed 'transfer' attempt from '0x5830d6a701c568545dcf8b3f8b75f56bed0c4' to '0x5830d6a701c568545dcf8b3f8b75f56bed0c4' with a value of 8000. The error message is: 'The transaction has been reverted to the initial state. Reason provided by the contract: "Insufficient balance". If the transaction failed for not having enough gas, try increasing the gas limit gently.'

```
15
16 constructor(uint256 _initialSupply) {
17     // TODO: Initialize totalSupply and assign it to the deployer's balance.
18     // HINT: Use _initialSupply and the 'decimals' variable.
19
20     require(_initialSupply > 0, "Value should be greater than 0");
21     totalSupply = _initialSupply;
22     balances[msg.sender] = _initialSupply * (10 ** decimals);
23 }
24
25 function balanceOf(address account) public view returns (uint256) {
26     // TODO: Return the balance of the given account.
27     // HINT: Use the 'balances' mapping.
28     return balances[account];
29 }
30
31 function transfer(address to, uint256 value) public returns (bool) {
32     require(balances[msg.sender] >= value, "Insufficient balance");
33     // TODO: Implement the transfer logic.
34     // HINT: Deduct 'value' from sender's balance and add it to the recipient's balance.
35
36     require(value > 0, "Value should be greater than 0");
37     balances[msg.sender] = balances[msg.sender] - value;
38     balances[to] = balances[to] + value;
39 }
40
```

2. TransferFrom: Insufficient Balance

The screenshot displays the Remix IDE interface with the ERC20Token.sol contract loaded. The contract code is visible in the main editor, showing functions for mint, burn, allowance, and transferFrom. The transaction history on the right shows several successful transactions, including a transferFrom that failed due to "Insufficient balance".

```

function mint(uint256 value) public {
    // TODO: Increase 'totalSupply' by 'value' and add 'value' to the caller's balance.
    require(value > 0, "Value should be greater than 0");
    totalSupply = totalSupply + value;
    balances[msg.sender] = balances[msg.sender] + value;
    emit Transfer(address(0), msg.sender, value);
}

function burn(uint256 value) public {
    // TODO: Decrease 'totalSupply' by 'value' and subtract 'value' from the caller's balance.
    require(value > 0, "Value should be greater than 0");
    totalSupply = totalSupply - value;
    balances[msg.sender] = balances[msg.sender] - value;
    emit Transfer(msg.sender, address(0), value);
}

function allowance(address owner, address spender) public view returns (uint256) {
    // TODO: Return the allowance set by 'owner' for 'spender'.
    return allowances[owner][spender];
}

function transferFrom(address from, address to, uint256 value) public returns (bool) {
    require(balances[from] >= value, "Insufficient balance");
    require(allowances[from][msg.sender] >= value, "Allowance exceeded");
    // TODO: Implement the logic for transferring tokens on behalf of another address.
    // HINT: Update the 'balances' and 'allowances' mappings.
    balances[from] = balances[from] - value;
    balances[to] = balances[to] + value;
    allowances[from][msg.sender] = allowances[from][msg.sender] - value;
    allowances[from][msg.sender] = value;
    emit Transfer(from, to, value);
    return true;
}

function mint(uint256 value) public {
    // TODO: Increase 'totalSupply' by 'value' and add 'value' to the caller's balance.
    require(value > 0, "Value should be greater than 0");
    totalSupply = totalSupply + value;
    balances[msg.sender] = balances[msg.sender] + value;
    emit Transfer(address(0), msg.sender, value);
}

```

Transaction history:

- Transaction to ERC20Token.transfer pending ...
- [vm] from: 0xA8B...35cb2 to: ERC20Token.transfer(address,uint256) 0xA8B...a872c value: 0 wei data: 0x90...00032 logs: 1 hash: 0x62f...05aa
- Transaction to ERC20Token.approve pending ...
- [vm] from: 0x5B3...edc4 to: ERC20Token.approve(address,uint256) 0xA8B...a872c value: 0 wei data: 0x95...0008c logs: 1 hash: 0x92...5e56
- Transaction to ERC20Token.transferFrom pending ...
- [vm] from: 0xA8B...35cb2 to: ERC20Token.transferFrom(address,address,uint256) 0xA8B...a872c value: 0 wei data: 0x23b...001f4 logs: 0 hash: 0x677...cd32
- Transaction to ERC20Token.transferFrom error: Error occurred: revert.

3. Allowance: Allowance exceeded.

The screenshot displays the Remix IDE interface with the ERC20Token.sol contract loaded. The contract code is visible in the main editor, showing functions for allowance, transferFrom, and mint. The transaction history on the right shows several failed transactions due to "Allowance exceeded".

```

function allowance(address owner, address spender) public view returns (uint256) {
    // TODO: Return the allowance set by 'owner' for 'spender'.
    return allowances[owner][spender];
}

function transferFrom(address from, address to, uint256 value) public returns (bool) {
    require(balances[from] >= value, "Insufficient balance");
    require(allowances[from][msg.sender] >= value, "Allowance exceeded");
    // TODO: Implement the logic for transferring tokens on behalf of another address.
    // HINT: Update the 'balances' and 'allowances' mappings.
    balances[from] = balances[from] - value;
    balances[to] = balances[to] + value;
    allowances[from][msg.sender] = allowances[from][msg.sender] - value;
    allowances[from][msg.sender] = value;
    emit Transfer(from, to, value);
    return true;
}

function mint(uint256 value) public {
    // TODO: Increase 'totalSupply' by 'value' and add 'value' to the caller's balance.
    require(value > 0, "Value should be greater than 0");
    totalSupply = totalSupply + value;
    balances[msg.sender] = balances[msg.sender] + value;
    emit Transfer(address(0), msg.sender, value);
}

```

Transaction history:

- Transaction to ERC20Token.transferFrom error: Error occurred: revert.
- Transaction to ERC20Token.transferFrom error: Error occurred: revert.
- Transaction to ERC20Token.transferFrom pending ...
- [vm] from: 0xA8B...35cb2 to: ERC20Token.transferFrom(address,address,uint256) 0xA8B...a872c value: 0 wei data: 0x23b...00014 logs: 0 hash: 0x3cb...0008
- Transaction to ERC20Token.transferFrom error: Error occurred: revert.
- Transaction to ERC20Token.transferFrom error: Error occurred: revert.