

Cs584 lab2

Peifeng He

Matseoi Zau

Task 1

- Deploy

The screenshot displays the Remix IDE interface during the deployment of a contract. The left sidebar contains the 'DEPLOY & RUN TRANSACTIONS' panel, which is configured with the following settings:

- ENVIRONMENT:** Remix VM (Cancun)
- ACCOUNT:** 0x5B3...eddC4 (99.999999999991662938 ether)
- GAS LIMIT:** Estimated Gas (selected), Custom (3000000)
- VALUE:** 0 Wei
- CONTRACT:** DukeCompsciToken - Token.sol
- DEPLOY:** SYMBOL: Compsci584, INITIALSUPPLY: 10. A 'transact' button is visible.
- Transactions recorded:** 174. A checkbox for 'Run transactions using the latest compilation result' is checked.
- Buttons:** 'Save' and 'Run' buttons are at the bottom.

The main editor shows the Solidity code for the `DukeCompsciToken` contract. The code includes a constructor, an `approve` function, and a `transferFrom` function. The right sidebar displays the transaction log, showing two calls to `AMMPool.getUserStatus()` and the creation of `DukeCompsciToken`.

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

contract DukeCompsciToken {
    uint256 private _totalSupply;
    string private _symbol;
    mapping(address => uint256) private balances;
    mapping(address => mapping(address => uint256)) private allowances;
    mapping(address => uint256) private lockedBalances;

    constructor(string memory symbol, uint256 initialSupply) {
        _symbol = symbol;
        _totalSupply = initialSupply;
        balances[msg.sender] = _totalSupply;
    }

    function approve(address spender, uint256 amount) public returns (bool) {
        require(amount <= balances[msg.sender] - lockedBalances[msg.sender], "Not enough unlocked");
        allowances[msg.sender][spender] += amount;
        lockedBalances[msg.sender] += amount; // Lock the approved amount
        return true;
    }

    function transferFrom(
        address from,
        address to,
        uint256 amount
    ) public returns (bool) {
        require(amount <= balances[from], "Insufficient balance");
        require(amount <= allowances[from][msg.sender], "Allowance exceeded");
    }
}
```

Task1

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

- DEPLOY & RUN TRANSACTIONS** (Left Panel):
 - INITIAL SUPPLY**: Set to 10.
 - Buttons**: Calculate, Parameters, Deploy.
 - Publish to IPFS**: Toggle switch.
 - At Address**: Field for loading a contract from an address.
 - Transactions recorded**: 1/15.
 - Run transactions using the latest compilation result**: Checked.
 - Save** and **Run** buttons.
 - Deployed Contracts**:
 - DUKECOMPSCITOKEN AT 0x577...2d343 (MEMORY)**:
 - Balance**: 0 ETH.
 - approve**: address spender, uint256 amount.
 - transfer**: 0xA0683F6A9C61Ecf96649Ae776D3315635d2_3.
 - transferFrom**: address from, address to, uint256 amount.
 - undo approve**: address spender, uint256 amount.
 - allowanceOf**: address account, address spender.
 - balanceOf**: 0xA0683F6A9C61Ecf96649Ae776D3315635d2.

- Editor** (Center):
- File Explorer**: Tokens.sol, Pool.sol, Introduction.sol, test.sol, scenario.json.
- Code**:

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.7.0 <0.9.0;
3
4 contract DukeCompsciToken {
5     uint256 private _totalSupply;
6     string private _symbol;
7     mapping(address => uint256) private balances;
8     mapping(address => mapping(address => uint256)) private allowances;
9     mapping(address => uint256) private lockedBalances;
10
11
12     constructor(string memory symbol, uint256 initialSupply) {
13         _symbol = symbol;
14         _totalSupply = initialSupply;
15         balances[msg.sender] = _totalSupply;
16     }
17
18     function approve(address spender, uint256 amount) public returns (bool) {
19         require(amount <= balances[msg.sender] - lockedBalances[msg.sender], "Not enough unlocked tokens");
20         allowances[msg.sender][spender] += amount;
21         lockedBalances[msg.sender] += amount; // Lock the approved amount
22         return true;
23     }
24
25     function transferFrom(
26         address from,
27         address to,
28         uint256 amount
29     ) public returns (bool) {
30         require(amount <= balances[from], "Insufficient balance");
31         require(amount <= allowances[from][msg.sender], "Allowance exceeded");
```
- Console** (Bottom):
- Log**: [call] from: 0x58380a6a701c568545dCfc883Fc8675f56beddC4 to: DukeCompsciToken.balanceOf(address) data: 0xa06...eddC4
- Log**: [vm] from: 0x583...eddC4 to: DukeCompsciToken.transfer(address,uint256) 0x577...2d343 value: 0 wei data: 0xa06...eddC4 logs: 0 hash: 0x54f...561ce

Balance of account
0x...C4:10.

Transfer 3 to 0x...b2

0x...b2 now have 3
tokens

Task1

The screenshot displays a Solidity IDE interface with a dark theme. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is active, showing the 'DUKECOMPSCITOKEN' contract deployed at address 0x377...2D343. The contract's balance is 0 ETH. Below this, a list of transactions is shown, including 'approve', 'transfer', 'transferFrom', 'undo_approve', 'allowancesOf', and 'balanceOf'. The 'balanceOf' transaction is selected, showing a call from 0x58380a6a701c568545dcfc803fc8875f56beddC4 to the contract's balanceOf function, with data 0x78a...35cb2. The main editor shows the Solidity code for the 'DukeCompsciToken' contract, which includes a constructor, an 'approve' function, and a 'transferFrom' function. The code is as follows:

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.7.0 <0.9.0;
3
4 contract DukeCompsciToken {
5     uint256 private _totalSupply;
6     string private _symbol;
7     mapping(address => uint256) private balances;
8     mapping(address => mapping(address => uint256)) private allowances;
9     mapping(address => uint256) private lockedBalances;
10
11
12     constructor(string memory symbol, uint256 initialSupply) {
13         _symbol = symbol;
14         _totalSupply = initialSupply;
15         balances[msg.sender] = _totalSupply;
16     }
17
18     function approve(address spender, uint256 amount) public returns (bool) {
19         require(amount <= balances[msg.sender] - lockedBalances[msg.sender], "Not enough unlocked");
20         allowances[msg.sender][spender] += amount;
21         lockedBalances[msg.sender] += amount; // Lock the approved amount
22         return true;
23     }
24
25     function transferFrom(
26         address from,
27         address to,
28         uint256 amount
29     ) public returns (bool) {
30         require(amount <= balances[from], "Insufficient balance");
31         require(amount <= allowances[from][msg.sender], "Allowance exceeded");
32     }
33 }
```

At the bottom, the transaction history shows two calls from 0x58380a6a701c568545dcfc803fc8875f56beddC4 to the contract's balanceOf function, with data 0x78a...35cb2. The first call is highlighted, and a 'Debug' button is visible next to it.

Only 7 left in 0x...C4

Task2

The screenshot displays a web3 development environment with three main panels:

- Left Panel (Deploy & Run Transactions):** Shows the 'Deployed Contracts' section for 'DUKECOMPSCITOKEN AT 0x577...2D343 (MEMORY)'. The balance is 0 ETH. The 'transferFrom' function is selected, showing a transaction from '0x5B38Da6a701c568545dCfcB03FcB875f56beddC4' to '0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db' with an amount of 2. The 'balanceOf' function is also visible, showing a balance of 0 for the same address.
- Center Panel (Solidity Code):** Displays the Solidity code for the 'DukeCompSciToken' contract. The code includes a constructor, an 'approve' function, and a 'transferFrom' function. The 'approve' function checks if the sender has enough unlocked balance and locks the approved amount. The 'transferFrom' function checks if the sender has enough balance and if the recipient has a valid allowance.
- Right Panel (Transaction Log):** Shows the execution of the 'transferFrom' function. The log entry indicates that the function was called with the address '0x5B38Da6a701c568545dCfcB03FcB875f56beddC4' and the amount '2'. The log also shows the state of the contract after the transaction, including the balance of the sender and the allowance of the recipient.

Transfer 2
tokens with
approval.
Check
balanceOf
0x...db, now
have two tokens
with approval.

Task3

The screenshot displays the Remix IDE interface in its initial state. The left sidebar contains a vertical menu with icons for deployment, search, and other functions. The main workspace is divided into three panels:

- DEPLOY & RUN TRANSACTIONS:** This panel on the left shows the deployment status. It includes a "Publish to IPFS" checkbox, a "transact" button, and a section for "Deployed Contracts". Under this section, two contracts are listed: "DUKECOMPSCITOKEN AT 0XA7D...6027B (MEMORY)" and "DUKECOMPSCITOKEN AT 0XEF9...96AB7 (MEMORY)". Below these, the "AMMPOOL AT 0XED3...69A05 (MEMORY)" contract is expanded, showing its "Balance: 0 ETH" and a list of functions: "addLiquidity", "swapXY", "getAllowance", "getPoolStatus", and "getUserStatus". Each function has a corresponding button and a "Transact" button at the bottom.
- Code Editor:** The central panel displays the Solidity code for the "Token.sol" file. The code defines an interface "DukeCompsciToken" with functions "transfer", "transferFrom", "balanceOf", and "allowancesOf". It then defines a contract "AMMPool" that implements these functions, including a constructor and a "swapXY" function.
- Console:** The bottom panel shows the execution log. It displays a call to "AMMPool.getUserStatus" with the following data: "[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: AMMPool.getUserStatus() data: 0xe65...0f536". A "Debug" button is visible next to the log entry.

Initial State

Check all
parameter
s at bottom
of the left
corner.

Task3

The screenshot displays a Solidity development environment. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is active, showing a list of functions: `transferFrom`, `undo_approve`, `allowancesOf`, and `balanceOf`. Below these, there are sections for 'Low level interactions' and 'CALLDATA'. The 'AMMPOOL AT 0XED3...69A05 (MEMORY)' section shows a balance of 0 ETH and several functions: `addLiquidity`, `swapXY`, `getAllowance`, `getPoolStatus`, and `getUserStatus`. The `getAllowance` function is selected, showing its parameters: `0: uint256: 1` and `1: uint256: 0`. The `getPoolStatus` function shows `0: uint256: 0` and `1: uint256: 2`. The `getUserStatus` function shows `0: uint256: 1` and `1: uint256: 0`. A 'Transact' button is visible at the bottom of the panel.

The right side of the image shows a code editor with the following Solidity code:

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity >=0.7.0 <0.9.0;
3
4 interface DukeCompSciToken {
5     function transfer(address recipient, uint256 amount) external returns (bool);
6     function transferFrom(address sender, address recipient, uint256 amount) external returns (bool);
7     function balanceOf(address account) external view returns (uint256);
8     function allowancesOf(address account, address spender) external view returns (uint256);
9 }
10
11 contract AMMPool {
12     DukeCompSciToken tokenX;
13     DukeCompSciToken tokenY;
14     event Debug(uint256 value);
15     event SwapExecuted(address indexed user, uint256 amountX, uint256 amountY, uint256 reserveX, uint256 reserveY);
16
17     constructor(address _tokenX, address _tokenY) {
18         tokenX = DukeCompSciToken(_tokenX);
19         tokenY = DukeCompSciToken(_tokenY);
20     }
21
22     function swapXY(uint256 amountX) public {
23         require(tokenX.allowancesOf(msg.sender, address(this)) >= amountX, "Approval required for tokenX");
24         require(tokenX.balanceOf(msg.sender) >= amountX, "Insufficient tokenX balance");
25
26         require(tokenX.transferFrom(msg.sender, address(this), amountX), "TokenX transfer failed");
27
28         uint256 reserveX = tokenX.balanceOf(address(this));
29         uint256 reserveY = tokenY.balanceOf(address(this));
30
31         // uint256 numerator = reserveY * amountX;
32         // uint256 denominator = reserveX + amountX;
33
34         uint256 amountY = 2*amountX;
35
36         require(reserveY >= amountY, "Insufficient tokenY liquidity");
37         require(tokenY.transfer(msg.sender, amountY), "TokenY transfer failed");
38     }
39 }
```

At the bottom of the code editor, a debug console shows a call to `AMMPool.getUserStatus` with the following data: `[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: AMMPool.getUserStatus() data: 0xe65...0f536`. A 'Debug' button is visible next to the call.

Get approved(has allowance)

A. approve (P, 1)

Task3

The screenshot displays the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel shows the deployment of the 'AMMPOOL' contract at address 0xED3...69A05. Below this, the 'swapXY' function is selected, and its parameters are set to 'amountX: 1'. The 'Transact' button is visible. The main editor shows the Solidity code for the 'AMMPOOL' contract, which implements the 'DukeCompSciToken' interface. The code includes functions for 'transfer', 'transferFrom', 'balanceOf', 'allowancesOf', 'swapXY', and 'getUserStatus'. The 'swapXY' function is highlighted, showing its logic for swapping tokens based on their reserves. The bottom panel shows a call to 'AMMPOOL.getUserStatus()' with the data '0xe65...0f536'.

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.7.0 <0.9.0;
3
4 interface DukeCompSciToken {
5     function transfer(address recipient, uint256 amount) external returns (bool);
6     function transferFrom(address sender, address recipient, uint256 amount) external returns (bool);
7     function balanceOf(address account) external view returns (uint256);
8     function allowancesOf(address account, address spender) external view returns (uint256);
9 }
10
11 contract AMMPool {
12     DukeCompSciToken tokenX;
13     DukeCompSciToken tokenY;
14     event Debug(uint256 value);
15     event SwapExecuted(address indexed user, uint256 amountX, uint256 amountY, uint256 reserveX, uint256 reserveY);
16
17     constructor(address _tokenX, address _tokenY) {
18         tokenX = DukeCompSciToken(_tokenX);
19         tokenY = DukeCompSciToken(_tokenY);
20     }
21
22     function swapXY(uint256 amountX) public {
23         require(tokenX.allowancesOf(msg.sender, address(this)) >= amountX, "Approval required for tokenX");
24         require(tokenX.balanceOf(msg.sender) >= amountX, "Insufficient tokenX balance");
25
26         require(tokenX.transferFrom(msg.sender, address(this), amountX), "TokenX transfer failed");
27
28         uint256 reserveX = tokenX.balanceOf(address(this));
29         uint256 reserveY = tokenY.balanceOf(address(this));
30
31         // uint256 numerator = reserveY * amountX;
32         // uint256 denominator = reserveX + amountX;
33
34         uint256 amountY = 2*amountX;
35
36
37         require(reserveY >= amountY, "Insufficient tokenY liquidity");
38         require(tokenY.transfer(msg.sender, amountY), "TokenY transfer failed");
39     }
40 }
```

Swapped.

The assets #
has been
updated

Task4

The screenshot displays a web3 development environment with a dark theme. On the left, a sidebar contains icons for various tools. The main area is divided into two panels. The top panel, titled 'DEPLOY & RUN TRANSACTIONS', shows a list of functions: 'transferFrom', 'undo_approve', 'allowancesOf', and 'balanceOf'. Below this, there's a section for 'Low level interactions' with a 'CALLDATA' input field and a 'Transact' button. The bottom panel shows the 'swapXY' function with an 'amountX' input field and a 'transact' button. The right panel displays the Solidity code for the 'AMMPool' contract, which includes an interface 'DukeCompsciToken' and functions 'transfer', 'transferFrom', 'balanceOf', 'allowancesOf', and 'swapXY'. The bottom of the right panel shows the execution logs, which include several debug messages and a final error message: 'The transaction has been reverted to the initial state. Reason provided by the contract: "Approval required for tokenX". If the transaction failed for not having enough gas, try increasing the gas limit gently.'

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity >=0.7.0 <0.9.0;
3
4 interface DukeCompsciToken {
5     function transfer(address recipient, uint256 amount) external returns (bool);
6     function transferFrom(address sender, address recipient, uint256 amount) external returns (bool);
7     function balanceOf(address account) external view returns (uint256);
8     function allowancesOf(address account, address spender) external view returns (uint256);
9 }
10
11 contract AMMPool {
```

creation of AMMPool pending...

[vm] from: 0x5B3...eddC4 to: AMMPool.(constructor) value: 0 wei data: 0x608...daef1 logs: 0 hash: 0x84b...76c11

transact to DukeCompsciToken.transfer pending ...

[vm] from: 0x5B3...eddC4 to: DukeCompsciToken.transfer(address,uint256) 0xA6D...daEF1 value: 0 wei data: 0xa90...00002 logs: 0 hash: 0x8bc...7a3d5

call to AMMPool.getAllowance

[call] from: 0x5B380a6a701c568545dCfcB03FcB875f56beddC4 to: AMMPool.getAllowance() data: 0x973...e9b8b

call to AMMPool.getPoolStatus

[call] from: 0x5B380a6a701c568545dCfcB03FcB875f56beddC4 to: AMMPool.getPoolStatus() data: 0x7f7...9496c

call to AMMPool.getUserStatus

[call] from: 0x5B380a6a701c568545dCfcB03FcB875f56beddC4 to: AMMPool.getUserStatus() data: 0xe65...0f536

transact to AMMPool.swapXY pending ...

[vm] from: 0x5B3...eddC4 to: AMMPool.swapXY(uint256) 0xcAB...C9e2f value: 0 wei data: 0x6db...00001 logs: 0 hash: 0xd0f...73cc5

transact to AMMPool.swapXY errored: Error occurred: revert.

revert

The transaction has been reverted to the initial state.
Reason provided by the contract: "Approval required for tokenX".
If the transaction failed for not having enough gas, try increasing the gas limit gently.

Without approval:

If swap them directly, an error will occur
“Approval required for tokenX”

Task4

The screenshot displays a web3 development environment with a Solidity contract named `DukeCompSciToken` and its deployment interface.

Contract Code (Token.sol):

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity >=0.7.0 <0.9.0;
3
4 contract DukeCompSciToken {
5     uint256 private _totalSupply;
6     string private _symbol;
7     bool ongoingApprove;
8     mapping(address => uint256) private balances;
9     mapping(address => mapping(address => uint256)) private allowances;
10    mapping(address => uint256) private lockedBalances;
11
12
13    constructor(string memory symbol, uint256 initialSupply) {
14        _symbol = symbol;
15        _totalSupply = initialSupply;
16        balances[msg.sender] = _totalSupply;
17        ongoingApprove = false;
18    }
19
20    function approve(address spender, uint256 amount) public returns (bool) {
21        require(amount <= balances[msg.sender] - lockedBalances[msg.sender], "Not enough unlocked tokens");
22        require(!ongoingApprove, "Another use is trying to transact, please wait!");
23
24        allowances[msg.sender][spender] += amount;
25        ongoingApprove = true;
26        lockedBalances[msg.sender] += amount; // Lock the approved amount
27    }
28
29
30    function transferFrom(
31        address from,
32        address to,
33        uint256 amount
34    ) public returns (bool) {
35        require(amount <= balances[from], "Insufficient balance");
36        require(amount <= allowances[from][msg.sender], "Allowance exceeded");
37
38        balances[from] -= amount;
39        balances[to] += amount;
40    }
41}
```

Deployment Interface (Left Panel):

- INITIALSUPPLY:** 5
- Buttons:** Calldata, Parameters, **transact**
- Publish to IPFS:** ☐
- At Address:** Load contract from Address
- Transactions recorded:** 10
- Deployed Contracts:** 1
 - DUKECOMPSCITOKEN AT 0x5C7...13682 (MEMORY)**
 - Balance: 0 ETH
 - approve**
 - spender: 0xdD870fA1b7C4700F2BD7f44238821C26f7392148
 - amount: 1
 - Buttons:** Calldata, Parameters, **transact**
 - transfer**: address receiver, uint256 amount
 - transferFrom**: address from, address to, uint256 amount
 - undo_approve**: address spender, uint256 amount
 - allowancesOf**: address account, address spender
 - balanceOf**: address account

Low level interactions:

- CALLDATA:**
- Transact**

Bottom Panel (Transaction Log):

- transact to DukeCompSciToken.approve pending ...
- [vm] from: 0x5B3...eddC4 to: DukeCompSciToken.approve(address,uint256) 0x5C7...13682 value: 0 wei data: 0x095...00001 logs: 0
- hash: 0x02d...753e2
- Buttons:** Debug

Approve a
certain
amount

Task4

The screenshot displays a web interface for deploying and running transactions. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel shows the initial supply set to 5. Below this, there are tabs for 'Calldata', 'Parameters', and 'transact'. The 'transact' tab is active, showing a transaction for the 'approve' function with a spender address and an amount of 1. Below the transaction details, there are buttons for 'transfer', 'transferFrom', 'undo_approve', 'allowancesOf', and 'balanceOf'. At the bottom, there is a 'Low level interactions' section with a 'Transact' button.

The main panel shows the Solidity code for the 'DukeCompSciToken' contract. The code is as follows:

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity >=0.7.0 <0.9.0;
3
4 contract DukeCompSciToken {
5     uint256 private _totalSupply;
6     string private _symbol;
7     bool ongoingApprove;
8     mapping(address => uint256) private balances;
9     mapping(address => mapping(address => uint256)) private allowances;
10    mapping(address => uint256) private lockedBalances;
11 }
```

On the right, a list of transactions is shown. The transactions are as follows:

- Transaction 1: [vm] from: 0xd08...92148 to: DukeCompSciToken.(constructor) value: 0 wei data: 0x608...0000 logs: 0 hash: 0xa02...389f3. Status: pending.
- Transaction 2: [vm] from: 0x5B3...eddC4 to: DukeCompSciToken.approve(address,uint256) 0x5D4...58DC1 value: 0 wei data: 0x095...0001 logs: 0 hash: 0xb60...4f27d. Status: pending.
- Transaction 3: [vm] from: 0xd08...92148 to: DukeCompSciToken.approve(address,uint256) 0x099...f7Fb9 value: 0 wei data: 0x095...0001 logs: 0 hash: 0x3b9...999d0. Status: pending.
- Transaction 4: [vm] from: 0x5B3...eddC4 to: DukeCompSciToken.(constructor) value: 0 wei data: 0x608...0000 logs: 0 hash: 0x5a0...569e5. Status: pending.
- Transaction 5: [vm] from: 0x5B3...eddC4 to: DukeCompSciToken.transfer(address,uint256) 0x5C7...13682 value: 0 wei data: 0xa90...0003 logs: 0 hash: 0x95a...fd1e8. Status: pending.
- Transaction 6: [vm] from: 0x5B3...eddC4 to: DukeCompSciToken.approve(address,uint256) 0x5C7...13682 value: 0 wei data: 0x095...0001 logs: 0 hash: 0x02d...753e2. Status: pending.
- Transaction 7: [vm] from: 0x5B3...eddC4 to: DukeCompSciToken.approve(address,uint256) 0x5C7...13682 value: 0 wei data: 0x095...0001 logs: 0 hash: 0x5f7...5f149. Status: errored. Reason: Error occurred: revert.

The error message for the last transaction is: "Another use is trying to transact, please wait!".

Another use trying to create another allowance and transact before the previous transaction happens:

Error: “Another user is trying to transact, please wait!”

Could not create a new approval.

Task5

The screenshot displays the Remix IDE interface, which is used for developing and testing smart contracts. The interface is divided into several panels:

- DEPLOY & RUN TRANSACTIONS:** This panel on the left contains a 'transfer' section with a 'receiver' field (0xfc79Aa9DfabF722A72643d5597a8911e481bAd08) and an 'amount' field (4). Below this are buttons for 'transferFrom', 'undo_approve', 'allowancesOf', and 'balanceOf'. The 'Low level interactions' section includes a 'CALLDATA' field and a 'Transact' button. At the bottom, there are buttons for 'addLiquidity', 'swapXY', 'getAllowance', 'getPoolStatus', and 'getUserStatus'.
- Solidity Code Editor:** The right panel shows the Solidity code for two contracts: 'DukeCompSciToken' and 'AMMPool'. The 'DukeCompSciToken' contract defines functions for 'transfer', 'transferFrom', 'balanceOf', and 'allowancesOf'. The 'AMMPool' contract defines functions for 'addLiquidity', 'swapXY', 'getAllowance', 'getPoolStatus', and 'getUserStatus'.
- Transaction Log:** The bottom panel displays a list of transactions and calls. Each entry includes a status (green checkmark), a description of the transaction (e.g., '[vm] from: 0x5B3...eddC4 to: DukeCompSciToken.(constructor) value: 0 wei data: 0x608...00000 logs: 0 hash: 0xb75...ea1f7'), and a 'Debug' button.

Initial state

Task5

The screenshot displays a web3 development environment with a Solidity contract and its transaction logs.

Left Panel: DEPLOY & RUN TRANSACTIONS

- transferFrom**: address from, address to, uint256 amount
- undo_approve**: address spender, uint256 amount
- allowancesOf**: address account, address spender
- balanceOf**: address account

Low level interactions

CALLDATA

AMMPOOL AT 0XFC7...BAD08 (MEMORY)

Balance: 0 ETH

- addLiquidity**: uint256 amountX, uint256 amountY
- swapXY**: uint256 amountX
- getAllowance**: 0: uint256: 1, 1: uint256: 0
- getPoolStatus**: 0: uint256: 1, 1: uint256: 4
- getUserStatus**: 0: uint256: 1, 1: uint256: 0

Low level interactions

CALLDATA

Right Panel: Solidity Contract

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity >=0.7.0 <0.9.0;
3
4 interface DukeCompSciToken {
5     function transfer(address recipient, uint256 amount) external returns (bool);
6     function transferFrom(address sender, address recipient, uint256 amount) external returns (bool);
7     function balanceOf(address account) external view returns (uint256);
8     function allowancesOf(address account, address spender) external view returns (uint256);
9 }
10
```

Transaction Logs

- [vm] from: 0x583...eddC4 to: AMMPool.constructor value: 0 wei data: 0x608...5f720 logs: 0 hash: 0x026...d3969
- [vm] from: 0x583...eddC4 to: DukeCompSciToken.transfer(address,uint256) 0x81f...D7863 value: 0 wei data: 0xa90...00001 logs: 0
- [vm] from: 0x583...eddC4 to: DukeCompSciToken.transfer(address,uint256) 0xd3b...5F720 value: 0 wei data: 0xa90...00004 logs: 0
- [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: AMMPool.getAllowance() data: 0x973...e9b8b
- [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: AMMPool.getPoolStatus() data: 0x7f7...9496c
- [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: AMMPool.getUserStatus() data: 0xe65...0f536
- [vm] from: 0x583...eddC4 to: DukeCompSciToken.approve(address,uint256) 0x81f...D7863 value: 0 wei data: 0x095...00001 logs: 0
- [call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: AMMPool.getAllowance() data: 0x973...e9b8b

Approval Created

Can see in the
“getAllowance”

A . approve (P , 1)

Task5

The screenshot shows a web-based Solidity development environment. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is active, displaying a list of functions: 'undo_approve', 'allowancesOf', 'balanceOf', and 'swapXY'. The 'swapXY' function is selected, showing its parameters and a 'transact' button. Below this, there are buttons for 'getAllowance', 'getPoolStatus', and 'getUserStatus'. The main area displays the Solidity code for 'Token.sol', which includes a pragma statement and several functions: 'transfer', 'transferFrom', 'balanceOf', and 'allowancesOf'. The code is highlighted with syntax coloring. On the right, a list of transactions is shown, including details like 'from', 'to', 'data', and 'hash'. The transactions are sorted by time, with the most recent at the top. The interface is dark-themed and includes a sidebar with various icons for navigation and development tools.

Final result

A. swapXY (1)

Task6

The screenshot displays a web3 development interface with three main sections:

- Left Panel (Deployed Contracts):** Shows a contract named "DUKECOMPSCITOKEN AT 0X9EC...DDE84 (MEMORY)". It includes a "Balance: 0 ETH" and a list of functions: "approve", "transfer", "transferFrom", and "undo_approve". Below this, the "allowancesOf" section shows a table with columns for "account" and "spender". The "spender" field is currently set to "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2".
- Center Panel (Smart Contract Code):** Displays the Solidity code for the "DukeCompSciToken" contract. The code includes a constructor, an "approve" function, and a "transferFrom" function. The "approve" function is highlighted, showing its logic for updating allowances and locking balances.
- Right Panel (Transaction Log):** Shows a list of transactions. The top transaction is a "CALL" from "0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db" to "DukeCompSciToken.balanceOf(address)". Below it, a "transact to DukeCompSciToken.approve" transaction is shown, with a "Debug" button.

Initiate the allowance

Task6

The screenshot displays a web-based Solidity IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel shows the 'DUKECOMPSCITOKEN' contract at address 0x9EC...DDE84 (MEMORY). The balance is 0 ETH. The 'approve' function is selected, and the 'amount' is set to 1. The 'allowancesOf' function is also visible. The main editor shows the Solidity code for the 'DukeCompSciToken' contract, including the constructor, 'approve', and 'transferFrom' functions. The bottom panel shows the transaction log, indicating that the 'approve' function was successfully executed, and the 'allowancesOf' function was called.

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity >=0.7.0 <0.9.0;
3
4 contract DukeCompSciToken {
5     uint256 private _totalSupply;
6     string private _symbol;
7     mapping(address => uint256) private balances;
8     mapping(address => mapping(address => uint256)) private allowances;
9     mapping(address => uint256) private lockedBalances;
10
11
12     constructor(string memory symbol, uint256 initialSupply) {
13         _symbol = symbol;
14         _totalSupply = initialSupply;
15         balances[msg.sender] = _totalSupply;
16     }
17
18     function approve(address spender, uint256 amount) public returns (bool) {
19         require(amount <= balances[msg.sender] - lockedBalances[msg.sender], "Not enough unlocked t
20         allowances[msg.sender][spender] += amount;
21         lockedBalances[msg.sender] += amount; // Lock the approved amount
22         return true;
23     }
24
25     function transferFrom(
26         address from,
27         address to,
28         uint256 amount
29     ) public returns (bool) {
30         require(amount <= balances[from], "Insufficient balance");
31         require(amount <= allowances[from][msg.sender], "Allowance exceeded");
32     }
33 }
```

Transaction Log:

- [vm] from: 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db to: DukeCompSciToken.approve(address,uint256) 0x9ec...dde84 value: 0 wei data: 0x095...00001 logs: 0 hash: 0x65a...fc7ed
- call to DukeCompSciToken.allowancesOf
- [call] from: 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db to: DukeCompSciToken.allowancesOf(address,address) data: 0xc5d...35cb2

Undo the approval

Task6

DEPLOY & RUN TRANSACTIONS

☒ Run transactions using the latest compilation result

Save **Run**

Deployed Contracts 1

▼ DUKECOMPSCITOKEN AT 0x9EC...DDE84 (MEMORY)

Balance: 0 ETH

approve 0xab8483f64d9c6d1ecf9b849ae677d3315835cb2, 1

transfer address receiver, uint256 amount

transferFrom address from, address to, uint256 amount

undo_approve

spender: 0xab8483f64d9c6d1ecf9b849ae677d3315835cb2

amount: 1

allowancesOf

account: 0x4820993bc481177ec7e8f571ceCaE8A9e22C02db

spender: 0xab8483f64d9c6d1ecf9b849ae677d3315835cb2

balanceOf 0x4820993bc481177ec7e8f571ceCaE8A9e22C02db

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity >=0.7.0 <0.9.0;
3
4 contract DukeCompSciToken {
5     uint256 private _totalSupply;
6     string private _symbol;
7     mapping(address => uint256) private balances;
8     mapping(address => mapping(address => uint256)) private allowances;
9     mapping(address => uint256) private lockedBalances;
10
11
12     constructor(string memory symbol, uint256 initialSupply) {
13         _symbol = symbol;
14         _totalSupply = initialSupply;
15         balances[msg.sender] = _totalSupply;
16     }
17
18     function approve(address spender, uint256 amount) public returns (bool) {
19         require(amount <= balances[msg.sender] - lockedBalances[msg.sender], "Not enough unlocked tokens");
20         allowances[msg.sender][spender] += amount;
21         lockedBalances[msg.sender] += amount; // Lock the approved amount
22         return true;
23     }
24
25     function transferFrom(
26         address from,
27         address to,
28         uint256 amount
29     ) public returns (bool) {
30         require(amount <= balances[from], "Insufficient balance");
31         require(amount <= allowances[from][msg.sender], "Allowance exceeded");
32     }
33 }
```

Transaction History

[call] from: 0x4820993bc481177ec7e8f571ceCaE8A9e22C02db to: DukeCompSciToken.allowancesOf(address,address) data: 0xc5d...35cb2 **Debug**

transact to DukeCompSciToken.undo_approve pending ...

[vm] from: 0x482...C02db to: DukeCompSciToken.undo_approve(address,uint256) 0x9ec...dde84 value: 0 wei data: 0x70c...00001 logs: 0 hash: 0x1b9...ac603 **Debug**

call to DukeCompSciToken.allowancesOf

Allowance -> 0:

The approval has been cancelled