**Синтаксические ошибки (С)**

1. **Ошибка в объявлении контракта**:
   * Было: contract Token are ERC20("CryptoMonster", "CMON").
   * Исправлено: contract Token is ERC20("CryptoMonster", "CMON").
2. **Ошибка в объявлении перечисления**:
   * Было: enums Role.
   * Исправлено: enum Role.
3. **Ошибка в объявлении переменной dec**:
   * Было: uint256 dec = 10\*\*decimals();яяяяя.
   * Исправлено: uint256 dec = 10\*\*decimals();.
4. **Ошибка в функции takeWhitelistRequest**:
   * Было: delete requests[\_index].
   * Исправлено: Использование pop для удаления элемента массива.
5. **Ошибка в функции sendRequestToWhitelist**:
   * Было: for(uint256 i = 0; i < requests.lenth; i++).
   * Исправлено: for(uint256 i = 0; i < requests.length; i++).
6. **Ошибка в функции buyToken**:
   * Было: payable(owner).transfer(value).
   * Исправлено: payable(owner).transfer(msg.value).
7. **Ошибка в функции stopPublicPhase**:
   * Было: \_transfer(pubProv, msg.sender, userMaps[pubProv].tokens).
   * Исправлено: \_transfer(pubProv, msg.sender, userMap[pubProv].publicTokens).
8. **Ошибка в функции transferToken**:
   * Было: if (sender == owner).
   * Исправлено: if (msg.sender == owner).
9. **Ошибка в функции changePublicPrice**:
   * Было: pubPrice = newPrice.
   * Исправлено: pubPrice = \_price.

**Логические ошибки (Л)**

1. **Ошибка в модификаторе AccessControl**:
   * Было: require(userMap[msg.sender].role != \_role, "\_\_\_\_\_\_\_\_\_").
   * Исправлено: require(userMap[msg.sender].role == \_role, "Доступ запрещен").
2. **Ошибка в функции signIn**:
   * Было: require(passwordMap[\_login] != \_password, "\_\_\_\_\_\_\_\_\_").
   * Исправлено: require(passwordMap[\_login] == keccak256(abi.encode(\_password)), "Неверный пароль").
3. **Ошибка в функции transferToken**:
   * Было: require(userMap[msg.sender].seedTokens < \_amount, "\_\_\_\_\_\_\_\_\_").
   * Исправлено: require(userMap[msg.sender].seedTokens >= \_amount, "Недостаточно seed CMON").
4. **Ошибка в функции signUp**:
   * Было: function signUp (string \_login, string memory \_password).
   * Исправлено: function signUp (string memory \_login, string memory \_password).
5. **Ошибка в функции buyToken**:
   * Было: require(\_amount / dec <= 5\_000, unicode"Максимальное кол-во - 5.000 CMON").
   * Исправлено: require(\_amount <= 5\_000 \* dec, unicode"Максимальное кол-во - 5.000 CMON").

**Сообщения об ошибках (П)**

1. **Ошибка в модификаторе AccessControl**:
   * Было: "\_\_\_\_\_\_\_\_\_".
   * Исправлено: "Доступ запрещен".
2. **Ошибка в функции signIn**:
   * Было: "\_\_\_\_\_\_\_\_\_".
   * Исправлено: "Неверный пароль".
3. **Ошибка в функции transferToken**:
   * Было: "\_\_\_\_\_\_\_\_\_".
   * Исправлено: "Недостаточно seed CMON".
4. **Ошибка в функции takeMyAllowance**:
   * Было: "\_\_\_\_\_\_\_\_".
   * Исправлено: "Вы не являетесь получателем".
5. **Ошибка в функции changePublicPrice**:
   * Было: Отсутствует сообщение об ошибке.
   * Исправлено: require(userMap[msg.sender].role == Role.publicProvider, "Только публичный провайдер может изменить цену").

### ****Оптимизация (О1)****

1. **Использование uint256 для времени**:
   * Время (block.timestamp) возвращает uint256, но в коде используется uint256 для всех временных переменных. Это избыточно, так как uint64 достаточно для хранения времени.
   * Исправлено: Заменить uint256 на uint64 для startTime, Time\_dif, privPhase, seedPhase.

### 1) ****Оптимизация (О2)****

Оптимизация функции getLifeTime:

Текущая реализация:

return block.timestamp + Time\_dif - startTime;

Оптимизация: Можно сохранить startTime + Time\_dif в отдельную переменную, чтобы избежать повторного вычисления.

Исправление:

uint256 public adjustedStartTime = startTime - Time\_dif;

function getLifeTime() public view returns(uint256){

return block.timestamp - adjustedStartTime;

}

1)**Уязвимость в функции signIn:**

Уязвимость: Функция возвращает данные пользователя без проверки пароля.

Риск: Злоумышленник может получить доступ к данным пользователя без авторизации.

Исправление: Добавить проверку пароля перед возвратом данных.

2)**Уязвимость в функции buyToken:**

Уязвимость: Не проверяется, что msg.value достаточно для покупки токенов.

Риск: Пользователь может купить токены без оплаты.

Исправление: Добавить проверку:

require(msg.value >= \_amount \* tokenPrice, "Not enough ether sent");

3)**Уязвимость в функции transferToken**:

Уязвимость: Не проверяется, что \_amount не превышает баланс отправителя.

Риск: Пользователь может перевести больше токенов, чем у него есть.

Исправление: Добавить проверку:

require(userMap[msg.sender].seedTokens >= \_amount, "Not enough tokens");

**Тестирование функции (Ф1)**

1)Тестирование функции buyToken:

Цель: Проверить, что функция корректно обрабатывает покупку токенов в разных фазах.

Шаги:

Установить время в подготовительную фазу.

Попытаться купить токены (должно вернуть ошибку).

Установить время в приватную фазу.

Попытаться купить токены без вайтлиста (должно вернуть ошибку).

Добавить пользователя в вайтлист.

Попытаться купить токены (должно пройти успешно).

Установить время в публичную фазу.

Попытаться купить токены (должно пройти успешно).

2)Тестирование функции transferToken:

Цель: Проверить, что функция корректно обрабатывает перевод токенов.

Шаги:

Перевести seed токены (должно пройти успешно).

Перевести больше seed токенов, чем есть на балансе (должно вернуть ошибку).

Перевести private токены (должно пройти успешно).

Перевести public токены (должно пройти успешно).