Контрольная работа №2

Вариант 1

Основная ссылка:
https://onlinetestpad.com/q63mj6y7dvl3i

Задание 1.

1. Что такое газ? Лимит газа? Оценка стоимости выполнения кода смарт-контракта

Gas – единицы, в которых измеряется стоимость выполнения операции.
Gas Limit – максимальное кол-во единиц gas'а, которое вы можете заплатить в транзакции (нужен, чтобы избежать бесконечного списывания единиц gas'а в цикле).
Стоимость операций зависит от:

- Сложности выполняемой операции (включая наличие циклов, создание и изменение переменных и т.п.);
- Нагруженности сети в момент транзакции.

2. Что делает смарт-контракт Ethereum таким особенным по сравнению с другими программами?

Ethereum – многоцелевая система (в отличии от биткоина), в которой можно написать не только контракты, но и децентрализованные приложения (однако использование функций будет иметь свою стоимость) с использованием языка Solidity. Из минусов – смарт-контракты Ethereum работают медленно.

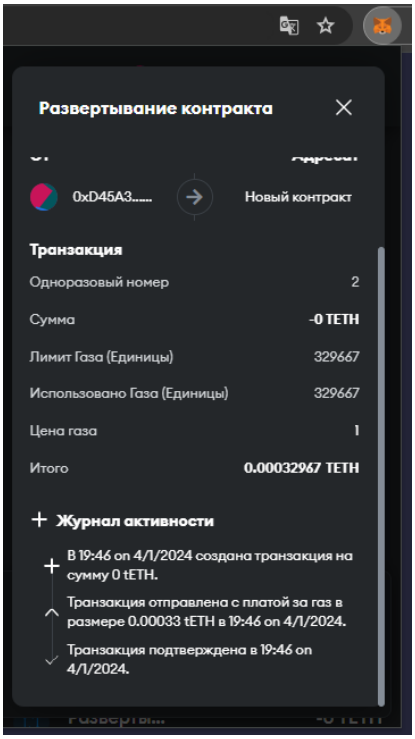
Задание 2.

Разработка смарт-контракта «Светофор» с функциями: 1) вывода информации о текущем сигнале светофора (красный, жёлтый, зелёный,), 2) функции установки сигнала светофора. Функция должна проверять, что если установлен красный, то следующий сигнал должен быть жёлтым, если жёлтый, то следующий зелёный, если зелёный, то следующий красный.

Выложить его в тестовую сеть и протестировать работу:

1) на JavaScript Virtual Machine,
2) в тестовой сети, записать время ожидания подтверждения транзакции создания смарт-контракта

```solidity
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.8.2 <0.9.0;

contract TrafficLights
{

    uint256 colorNumber;
    mapping(uint => string) colors;

    constructor()    infinite gas 182400 gas
    {
        colorNumber = 0;

        colors[0] = "Red";
        colors[1] = "Yellow";
        colors[2] = "Green";
    }

    function GetInfo() public view returns (string memory)    infinite gas
    {
        return colors[colorNumber];
    }

    function SetInfo() public returns (bool success)    infinite gas
    {
        colorNumber = colorNumber % 3 +1;
        return true;
    }
}
```



Задание 3

Смарт-контракт для децентрализованного благотворительного фонда

Описание задачи:

Разработать смарт-контракт на Ethereum для децентрализованного благотворительного фонда. Смарт-контракт должен позволять пользователям отправлять ETH на контракт в качестве пожертвований. Владелец контракта (администратор фонда) может перевести собранные средства на указанные благотворительные организации (адреса). Дополнительно, контракт должен предоставлять информацию о суммарных пожертвованиях и перечне организаций-получателей.

Требования к функционалу:

Функция пожертвования: Позволяет любому пользователю отправить ETH на контракт. При отправке средств пользователь может указать, в честь какого события или цели он делает пожертвование.

Функция перевода средств: Владелец контракта может перевести накопленные средства на другой счет (адрес организации). Данная функция должна быть защищена модификатором, ограничивающим вызов только владельцем.

Отчетность: Контракт должен вести историю пожертвований и переводов, позволяя просматривать общее количество пожертвований и список получателей.

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.7.12 <0.9.0;

contract FundForDonations
{
    address payable owner;              //адрес владельца контракта
    Tranfer[] public trasferReports;    //массив отчетов по пожертвованиям

    struct Tranfer                      //структура трансфер
    {
        address sender;                 //отправитель
        address organization;           //организация-получатель
        uint amount;                    //сумма
    }

    constructor()
    {
        owner = payable (msg.sender);
    }

    function Donate () payable public     414296 gas 393600 gas
    {
        require(msg.value > 0, "No money");
        owner.transfer(msg.value);    infinite gas
        trasferReports.push(Tranfer(msg.sender, owner, msg.value));
    }

    function SendFunds (address payable fund) payable public
    {
        require(msg.sender == owner, "You not owner");    undefined gas
        require(msg.value > 0, "No money");
        fund.transfer(msg.value);
        trasferReports.push(Tranfer(owner, fund, msg.value));
    }
}
```