

Работа победителя/призера заключительного этапа
командной инженерной олимпиады школьников
Олимпиада Национальной технологической инициативы

Профиль «Программная инженерия финансовых технологий»

Федосеев Кирилл Сергеевич

Класс: 11

Город: Краснообск

Школа: МАОУ Лицей №13

Регион: Новосибирская область

Уникальный номер участника: 4

Команда на заключительном этапе: Че пацаны, финтех?

Параллель: 10-11 класс

Результаты заключительного этапа:

№	индивидуальная			Командная часть				результат
	Математика	Информатика	итого	1 день	2 день	3 день	общий	
4	16	100	116	28	57	32	117	233

Индивидуальная часть

Персональный лист участника с номером 4:



Олимпиада НТИ

ФИО Ледосеев Кирилл Сергеевич

Город Новосибирск

Школа № Линей № 13

Математика

Лист 1:

Командная инженерная олимпиада «Олимпиада НТИ»

Направление Физика

Предмет Математика

Номер участника 1802016

№1
№2

а) 0 может появиться 2 способами $\left\{ \begin{array}{l} 0 \rightarrow 0 (P = p(1-\epsilon)) \\ \emptyset \rightarrow 0 (P = (1-p)\epsilon) \end{array} \right.$

$$P(\text{число } B_i \text{ равно } 0) = p(1-\epsilon) + (1-p)\epsilon = p + \epsilon - 2\epsilon p$$

$$P(\text{число } B_i \text{ равно } 1) = 1 - p - \epsilon + 2\epsilon p$$

б) $Q = \max_x H(B)$

$Q \rightarrow \max$, при $P_{B:0} = P_{B:1} = 0,5$

$$Q = H\left(\left[\frac{1}{2}; \frac{1}{2}\right]\right) = 1$$

Ответ: а) $p + \epsilon - 2\epsilon p$; б) $1 - p - \epsilon + 2\epsilon p$
 Ответ: а) 1

№3

а) Пусть $\nu = n - 1$.

$$(n-1)^3 \bmod n = ?$$

~~$$(n-1)^3 = (n-2)(n-1)(n-1)$$~~

$$(n-1)^3 + 1 = (n-1+1) \cdot \dots$$

$$(n-1)^3 + 1 \bmod n = 0$$

$$(n-1)^3 \bmod n = n - 1$$

Множество чисел n делителей, т.к. множество чисел n делителей

Ответ: ∞

б) Пусть $n = b + 1$

$$b^5 \bmod (b+1) = ?$$

$$(b^5 - b) \bmod (b+1) = 0, \text{ т.к. } b^5 - b = b(b-1)(b+1)(b^2+1)$$

$b^5 \bmod (b+1) = b$. \Rightarrow аналогично примет a ; Ответ: ∞

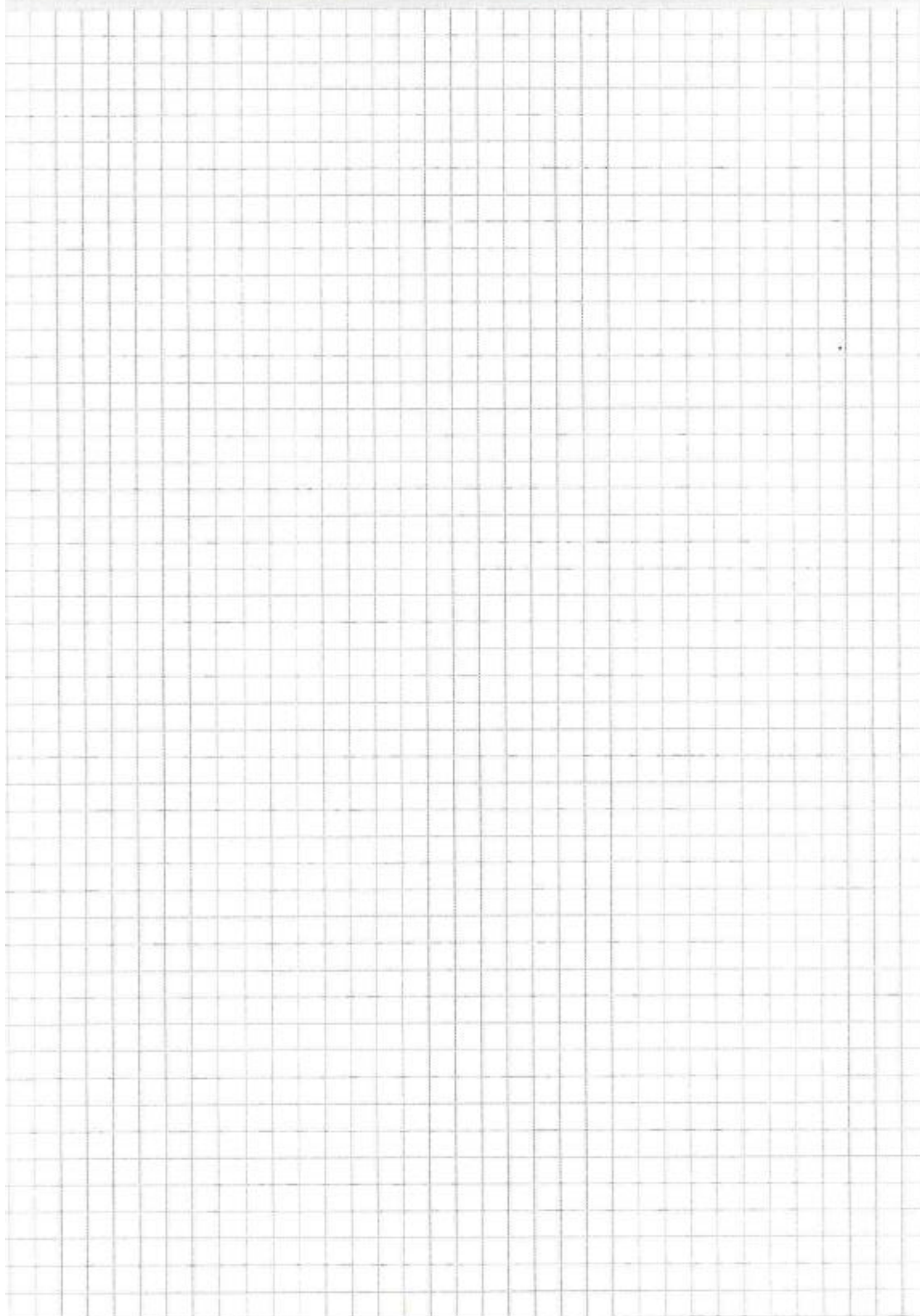
Лист 2:

Командная инженерная олимпиада «Олимпиада НТИ»

Направление Физика

Предмет Математика

Номер участника 1802015



Информатика

Задача A1:

```
#include <iostream>

#include <vector>

using namespace std;

vector<int> g[100010];
bool used[100010];

int dfs(int v) {
    used[v] = true;
    for(int i = 0; i < g[v].size(); ++i) {
        if(!used[g[v][i]])
            dfs(g[v][i]);
    }
    return 0;
}

int main() {
    int n, m, a, b;
    cin >> n >> m;
    for(int i = 0; i < m; ++i) {
        cin >> a >> b;
        g[a].push_back(b);
    }
    cin >> a >> b;
    dfs(a);
}
```

```
        if(used[b])
            cout << "Yes" << endl;
        else
            cout << "No" << endl;
        return 0;
    }
```

Задача A2:

```
#include <iostream>
#include <vector>
```

```
using namespace std;
```

```
vector<int> g[100010];
```

```
bool used[100010];
```

```
int dfs(int v) {
    used[v] = true;
    for(int i = 0; i < g[v].size(); ++i) {
        if(!used[g[v][i]])
            dfs(g[v][i]);
    }
    return 0;
}
```

```
int main() {
    int n, m, a, b;
    cin >> n >> m;
    for(int i = 0; i < m; ++i) {
```

```

        cin >> a >> b;
        g[a].push_back(b);
    }
    cin >> a >> b;
    dfs(a);
    if(used[b])
        cout << "Yes" << endl;
    else
        cout << "No" << endl;
    return 0;
}

```

Задача A Full:

```

#include <iostream>
#include <vector>

using namespace std;

vector<int> g[100010];
bool used[100010];

int dfs(int v) {
    used[v] = true;
    for(int i = 0; i < g[v].size(); ++i) {
        if(!used[g[v][i]])
            dfs(g[v][i]);
    }
    return 0;
}

```

```

int main() {
    int n, m, a, b;
    cin >> n >> m;
    for(int i = 0; i < m; ++i) {
        cin >> a >> b;
        g[a].push_back(b);
    }
    cin >> a >> b;
    dfs(a);
    if(used[b])
        cout << "Yes" << endl;
    else
        cout << "No" << endl;
    return 0;
}

```

Задача B1:

```

#include <iostream>
#include <vector>
#include <algorithm>
#include <queue>

using namespace std;

vector<pair<int, int> > g[100010];
bool used[100010];
vector<int> d(100010, 2000000000);
priority_queue<pair<int, int>> pq;

```



```

int main() {
    int n, m, a, b, c, s, f, q;
    cin >> n >> m;
    for (int i = 0; i < m; ++i) {
        cin >> a >> b >> c;
        g[a].push_back(make_pair(b, c));
    }
    cin >> s >> f;
    cin >> m >> q;
    for (int i = 0; i < m; ++i) {
        cin >> a >> b >> c;
        g[a].push_back(make_pair(b, c + q));
    }
    d[s] = 0;
    pq.push(make_pair(0, s));
    while (!pq.empty()) {
        pair<int, int> top = pq.top();
        pq.pop();
        int from = top.second;
        for (pair<int, int> &p : g[from]) {
            int to = p.first, cost = p.second;
            if (d[from] + cost < d[to]) {
                d[to] = d[from] + cost;
                pq.push(make_pair(d[to], to));
            }
        }
    }
}

```

```
    if(d[f] != 2000000000)
        cout << d[f] << endl;
    else
        cout << -1;
    return 0;
}
```

Задача B2:

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <queue>

using namespace std;

vector<pair<int, int> > g[100010];
bool used[100010];
vector<int> d(100010, 2000000000);
priority_queue<pair<int, int>> pq;

int main() {
    int n, m, a, b, c, s, f, q;
    cin >> n >> m;
    for (int i = 0; i < m; ++i) {
        cin >> a >> b >> c;
        g[a].push_back(make_pair(b, c));
    }
    cin >> s >> f;
    cin >> m >> q;
```

```

for (int i = 0; i < m; ++i) {
    cin >> a >> b >> c;
    g[a].push_back(make_pair(b, c + q));
}
d[s] = 0;
pq.push(make_pair(0, s));
while (!pq.empty()) {
    pair<int, int> top = pq.top();
    pq.pop();
    int from = top.second;
    for (pair<int, int> &p : g[from]) {
        int to = p.first, cost = p.second;
        if (d[from] + cost < d[to]) {
            d[to] = d[from] + cost;
            pq.push(make_pair(d[to], to));
        }
    }
}
if(d[f] != 2000000000)
    cout << d[f] << endl;
else
    cout << -1;
return 0;
}

```

Задача B Full:

```

#include <iostream>
#include <vector>
#include <algorithm>

```

```
#include <queue>

using namespace std;

vector<pair<int, int> > g[100010];
bool used[100010];
vector<int> d(100010, 2000000000);
priority_queue<pair<int, int>> pq;

int main() {
    int n, m, a, b, c, s, f, q;
    cin >> n >> m;
    for (int i = 0; i < m; ++i) {
        cin >> a >> b >> c;
        g[a].push_back(make_pair(b, c));
    }
    cin >> s >> f;
    cin >> m >> q;
    for (int i = 0; i < m; ++i) {
        cin >> a >> b >> c;
        g[a].push_back(make_pair(b, c + q));
    }
    d[s] = 0;
    pq.push(make_pair(0, s));
    while (!pq.empty()) {
        pair<int, int> top = pq.top();
        pq.pop();
        int from = top.second;
```

```

    if (!used[from]) {
        used[from] = true;
        for (pair<int, int> &p : g[from]) {
            int to = p.first, cost = p.second;
            if (d[from] + cost < d[to]) {
                d[to] = d[from] + cost;
                pq.push(make_pair(-d[to], to));
            }
        }
    }
}

if(d[f] != 2000000000)
cout << d[f] << endl;
else
cout << -1;
return 0;
}

```

Задача C:

```

#include <iostream>

#include <vector>

#include <algorithm>

#include <queue>

#include <map>

using namespace std;

vector<pair<int, int> > g[100010];

vector<bool> used(100010, false);

vector<int> val(100010, 0);

```

```
map<int, int> mp;
```

```
int kek;
```

```
int _dfs(int v, int x = 0) {  
    used[v] = true;  
    if (mp[x])  
        mp[x]++;  
    else  
        mp[x] = 1;  
    val[v] = x;  
    for (pair<int, int> &p : g[v]) {  
        if (!used[p.first]) {  
            _dfs(p.first, x ^ p.second);  
        }  
    }  
    return 0;  
}
```

```
int main() {  
    cin.sync_with_stdio(false);  
    cin.tie(0);  
    int n, q, a, b, c;  
    cin >> n >> q;  
    for (int i = 0; i < n - 1; i++){  
        cin >> a >> b >> c;  
        g[a].push_back(make_pair(b, c));  
        g[b].push_back(make_pair(a, c));  
    }
```

```
}  
_dfs(a);  
for (int i = 0; i < q; ++i) {  
    cin >> a >> kek;  
    if (mp[kek ^ val[a]])  
        cout << mp[kek ^ val[a]] << endl;  
    else  
        cout << 0 << endl;  
}  
}
```

Командная часть

Результаты были получены в рамках выступления команды: Че пацаны, финтех?

Фотография команды за работой:



Личный состав команды:

- Колпаков Артем Андреевич
- Федосеев Кирилл Сергеевич

Протокол выполнения заданий:

Первый этап:

№ задания	Критерий	Возможные баллы	Полученные баллы
US-001	Регистрация аккаунта компании производителя ПО	2	2
US-002	Регистрация контрактов	12	12
US-005	Создание аккаунта производителя батарей	2	2
US-007	Регистрация нового производителя (только АС-007-01)	8	0
US-010	Поштучная регистрация производимых батарей	6	0

US-020	Создание аккаунта сервисного центра	2	2
US-021	Регистрация аккаунта сервисного центра	6	6
US-022	Создание аккаунта электромобиля	2	2
US-023	Получение аккаунта электромобиля	2	2
US-024	Регистрация аккаунта электромобиля	6	0
Всего за этап		48	28

Второй этап:

№ задания	Критерий	Возможные баллы	Полученные баллы
US-003	Установка сборов за регистрацию одной батареи	6	6
US-006	Получение информации о необходимом депозите за регистрацию производителя батарей	4	4
US-007	Регистрация нового производителя (все)	4	4
US-008	Защита от повторной регистрации нового производителя	4	4
US-009	Получение информации о сборе за регистрацию одной батареи	4	4
US-011	Регистрация батарей с закрепленным сбором за регистрацию батарей	4	4
US-012	Получение размера остатка по депозиту	4	4
US-013	Пакетная регистрация производимых батарей	6	6
US-015	Поштучная продажа новых батарей	6	6
US-017	Генерация прошивок аппаратных ключей батарей	4	4
US-018	Изменение количества заряда батареи	3	3
US-019	Получение информации о батарее из прошивки	8	8
US-025	Проверка подлинности батареи сервисным центром	9	0
US-026	Проверка подлинности батареи электромобилем	9	0
US-027	Получение адреса контракта для замены батареи	16	0
US-029	Получение информации, что контракт замены в ожидании подтверждения	4	0
US-030	Получение условий контракта	8	0
US-031	Подтверждение согласия с условиями контракта	16	0
US-032	Подтверждение готовности к физической замене батарей	4	0
US-033	Подтверждение контракта	16	0
Всего за этап		139	57

Третий этап:

№ задания	Критерий	Возможные баллы	Полученные баллы
US-004	Регистрация контрактов с постоянными адресами	20	0
US-014	Безопасная пакетная регистрация производимых батарей	6	6
US-016	Пакетная продажа новых батарей	6	6
US-028	Отправка запроса на отмену сделки	8	0
US-034	Установка времени запрета разблокировки токенов	6	0
US-035	Запрос на разблокирование токенов	10	0

US-036	Отсутствие подтверждения замены батарей для завершения сделки	10	0
US-037	метод setBatteryManagementContract()	2	2
US-038	метод registerVendor()	4	4
US-039	метод registerBatteries()	3	3
US-040	метод setFee()	2	2
US-041	метод registerServiceCenter()	2	2
US-042	метод registerCar()	2	2
US-043	метод createBattery()	3	3
US-044	метод transfer() с указанием одного идентификатора батареи	2	2
US-045	метод delegatedApprove()	4	0
US-046	метод initiateDeal()	6	0
US-047	метод setTimeoutThreshold()	2	0
US-048	метод agreeToDeal()	6	0
US-050	метод releaseTokensByCar()	4	0
US-051	метод releaseTokensByServiceCenter()	4	0
US-052	метод cancelDeal()	4	0
US-053	Передача прав на батарею новому владельцу	8	0
US-054	Отложенная передача прав владения	12	0
US-055	Регистрация прав владения	8	0
US-056	Оптимизация по использованию вычислительных ресурсов	33	0
US-057	Оптимизация комиссии за валидацию транзакций	6	0
US-058	Сбор статистики	30	0
Всего за этап		213	32

Программа команды для решения задач:

Задача Car:

```

from utils import compile, wait_tx_receipt
from web3 import Web3
import os
import sys
import time
from db import write_car, write_database, read_car, read_database
from web3.middleware import geth_poa_middleware
import warnings
import random
import string
import subprocess

warnings.simplefilter("ignore", category=DeprecationWarning)

web3 = Web3()
web3.middleware_stack.inject(geth_poa_middleware, layer=0)

account = read_car()
database = read_database()

if account.get('account'):

```

```

        web3.personal.unlockAccount(account['account'],
account['password'])

command = sys.argv[1]

# US-022
if command == '--new':
    res = web3.eth.account.create()
    write_car({'key': res.privateKey.hex()[2:]})
    print(res.address)

# US-023
elif command == '--account':
    print(web3.eth.account.privateKeyToAccount(bytes.fromhex(account['
key'])).address)

# US-024
elif command == '--reg':
    abi = compile()[0]['abi']

    contract = web3.eth.contract(abi=abi,
address=database['mgmtContract'])

    acc =
web3.eth.account.privateKeyToAccount(bytes.fromhex(account['key']))
    try:
        tx = contract.functions.registerCar() \
            .buildTransaction({
                'nonce': web3.eth.getTransactionCount(acc.address)
            })

        signed_txn = web3.eth.account.signTransaction(tx,
private_key=acc.privateKey)
        web3.eth.sendRawTransaction(signed_txn.rawTransaction)
        tx_hash = web3.toHex(web3.sha3(signed_txn.rawTransaction))
        tx_receipt = wait_tx_receipt(web3, tx_hash)
        if tx_receipt['status'] == 1:
            print('Registered successfully')
        else:
            print('Already registered')
    except ValueError:
        print('Already registered')
elif command == '--verify':
    mgmt, _, battery_mgmt = compile()[3]

    contract = web3.eth.contract(abi=mgmt['abi'],
address=database['mgmtContract'])
    battery_contract = web3.eth.contract(abi=battery_mgmt['abi'],
address=contract.functions.batteryManagement().call())

    res = subprocess.check_output(['python', f'./{sys.argv[2]}', '--
get'])
    _res = res.split()
    n = int(_res[0])
    t = int(_res[1])
    v = int(_res[2])
    r = bytes.fromhex(_res[3].decode(encoding='utf-8').zfill(64))
    s = bytes.fromhex(_res[4].decode(encoding='utf-8').zfill(64))
    status, addr = battery_contract.functions.verifyBattery(n, t, v,
r, s).call()
    if status == 0:

```

```

        print('Verified successfully.')
        print(f'Total charges: {n}')
        id = contract.functions.vendorId(addr).call()
        print(f'Vendor ID: {id.hex()}')
        print(f"Vendor Name:
'{contract.functions.vendorNames(id).call().decode(encoding='utf-8')}'")
        elif status == 2:
            print('Verification failed. Probably the battery forged.')
        else:
            print('Battery with the same status already replaced.
Probably the battery forged.')

elif command == 'owner':
    mgmt, _, battery_mgmt = compile()[:3]

    contract = web3.eth.contract(abi=mgmt['abi'],
address=database['mgmtContract'])
    battery_contract = web3.eth.contract(abi=battery_mgmt['abi'],
address=contract.functions.batteryManagement().call())

    acc =
web3.eth.account.privateKeyToAccount(bytes.fromhex(account['key']))

    battery_contract.functions.transfer().buildTransaction({
        'chainId': 33,

    })

```

Задача db:

```

import json
import os

def write_account(acc=None):
    if acc is None:
        acc = {}
    with open('./account.json', 'w') as outfile:
        json.dump(acc, outfile)
    return acc

def write_scenter(acc=None):
    if acc is None:
        acc = {}
    with open('./scenter.json', 'w') as outfile:
        json.dump(acc, outfile)
    return acc

def write_car(acc=None):
    if acc is None:
        acc = {}
    with open('./car.json', 'w') as outfile:
        json.dump(acc, outfile)
    return acc

def write_database(db=None):

```

```

    if db is None:
        db = {}
    with open('./database.json', 'w') as outfile:
        json.dump(db, outfile)
    return db

def read_account():
    if os.path.exists('./account.json'):
        with open('./account.json') as file:
            return json.load(file)
    return {}

def read_scenter():
    if os.path.exists('./scenter.json'):
        with open('./scenter.json') as file:
            return json.load(file)
    return {}

def read_car():
    if os.path.exists('./car.json'):
        with open('./car.json') as file:
            return json.load(file)
    return {}

def read_database():
    if os.path.exists('./database.json'):
        with open('./database.json') as file:
            return json.load(file)
    return {}

def create_firmware(battery_id, key):
    if not os.path.exists('./firmware'):
        os.mkdir('./firmware')
    with open(f'./firmware/{battery_id[:8]}.py', 'w') as outfile:
        outfile.write(
            f'''
import json, os, time, sys
import eth_account.account as eth_account

_privkey = {key}

def charge():
    db_file = __file__[:-2] + '.json'
    if not os.path.exists(db_file):
        with open(db_file, 'w') as outfile:
            outfile.write('{chr(123)}"charges": 1{chr(125)}')
    else:
        with open(db_file, 'r') as file:
            data = json.load(file)
            data['charges'] += 1
            with open(db_file, 'w') as file:
                json.dump(data, file)

def get():
    data = {chr(123)}'charges': 0{chr(125)}

```

```

        db_file = __file__[:-2] + 'json'
        if os.path.exists(db_file):
            with open(db_file, 'r') as file:
                data = json.load(file)
            n = data['charges']
            t = round(time.time())
            signed = eth_account.Account.sign(message=((n << 32) +
t).to_bytes(32, byteorder='big'), private_key=_privkey)
            return (n, t, signed['v'], hex(signed['r'])[2:],
hex(signed['s'])[2:])

if sys.argv[1] == '--charge':
    charge()
elif sys.argv[1] == '--get':
    n, t, v, r, s = get()
    print(n)
    print(t)
    print(v)
    print(r)
    print(s)
'''

```

Задача Scenter:

```

from utils import compile, wait_tx_receipt
from web3 import Web3
import os
import sys
import time
from db import write_account, write_database, read_account, read_database,
read_scenter, write_scenter
from web3.middleware import geth_poa_middleware
import warnings
import subprocess

warnings.simplefilter("ignore", category=DeprecationWarning)

web3 = Web3()
web3.middleware_stack.inject(geth_poa_middleware, layer=0)

account = read_scenter()
database = read_database()

if account.get('account'):
    web3.personal.unlockAccount(account['account'],
account['password'])

command = sys.argv[1]

# US-020
if command == '--new':
    res = web3.personal.newAccount(sys.argv[2])
    print(res)
    write_scenter({'account': res, 'password': sys.argv[2]})

# US-021
elif command == '--reg':
    abi = compile()[0]['abi']

```

```

        contract = web3.eth.contract(abi=abi,
address=database['mgmtContract'])

        if not
contract.functions.serviceCenters(account['account']).call():
            tx_hash =
contract.functions.registerServiceCenter().transact({'from':
account['account']})
            tx_receipt = wait_tx_receipt(web3, tx_hash)
            print('Registered successfully')
        else:
            print('Already registered')

elif command == '--verify':
    mgmt, _, battery_mgmt = compile()[:3]

    contract = web3.eth.contract(abi=mgmt['abi'],
address=database['mgmtContract'])
    battery_contract = web3.eth.contract(abi=battery_mgmt['abi'],
address=contract.functions.batteryManagement().call())

    res = subprocess.check_output(['python', f'./{sys.argv[2]}', '--
get'])
    _res = res.split()
    n = int(_res[0])
    t = int(_res[1])
    v = int(_res[2])
    r = bytes.fromhex(_res[3].decode(encoding='utf-8').zfill(64))
    s = bytes.fromhex(_res[4].decode(encoding='utf-8').zfill(64))
    status, addr = battery_contract.functions.verifyBattery(n, t, v,
r, s).call()
    if status == 0:
        print('Verified successfully.')
        print(f'Total charges: {n}')
        id = contract.functions.vendorId(addr).call()
        print(f'Vendor ID: {id.hex()}')
        print(f"Vendor Name:
'{contract.functions.vendorNames(id).call().decode(encoding='utf-8')}'")
    elif status == 2:
        print('Verification failed. Probably the battery forged.')
    else:
        print('Battery with the same status already replaced.
Probably the battery forged.')

```

Задача Setup:

```

from utils import compile, wait_tx_receipt
from web3 import Web3
import os
import sys
import time
from db import write_account, write_database, read_account, read_database
from web3.middleware import geth_poa_middleware
import warnings

warnings.simplefilter("ignore", category=DeprecationWarning)

web3 = Web3()

```

```

web3.middleware_stack.inject(geth_poa_middleware, layer=0)

account = read_account()
database = read_database()

if account.get('account'):
    web3.personal.unlockAccount(account['account'],
account['password'])

command = sys.argv[1]

# US-001
if command == '--new':
    res = web3.personal.newAccount(sys.argv[2])
    print(res)
    write_account({'account': res, 'password': sys.argv[2]})

# US-002
elif command == '--setup':
    fee = web3.toWei(float(sys.argv[2]), 'ether')

    compiled_mgmt, compiled_wallet, compiled_bat, compiled_erc =
compile()

    management_contract = web3.eth.contract(abi=compiled_mgmt['abi'],
bytecode=compiled_mgmt['bin'])
    wallet_contract = web3.eth.contract(abi=compiled_wallet['abi'],
bytecode=compiled_wallet['bin'])
    battery_contract = web3.eth.contract(abi=compiled_bat['abi'],
bytecode=compiled_bat['bin'])
    erc_contract = web3.eth.contract(abi=compiled_erc['abi'],
bytecode=compiled_erc['bin'])

    wallet_contract_address = wait_tx_receipt(web3,
wallet_contract.deploy({'from': account['account']}))['contractAddress']
    erc_contract_address = wait_tx_receipt(web3,
erc_contract.deploy({'from': account['account']}))['contractAddress']

    management_contract_address = wait_tx_receipt(
web3,
management_contract.deploy({'from': account['account']},
[wallet_contract_address, fee])
)['contractAddress']

    battery_contract_address = wait_tx_receipt(
web3,
battery_contract.deploy({'from': account['account']},
[management_contract_address, erc_contract_address])
)['contractAddress']

    tx_hash =
management_contract.functions.setBatteryManagementContract(battery_contra
ct_address) \
    .transact({'from': account['account'], 'to':
management_contract_address})
    wait_tx_receipt(web3, tx_hash)

    print(f'Management contract: {management_contract_address}')
    print(f'Wallet contract: {wallet_contract_address}')
    print(f'Currency contract: {erc_contract_address}')

```



```

        write_database({'mgmtContract': management_contract_address})

# US-003
elif command == '--setfee':
    fee = web3.toWei(float(sys.argv[2]), 'ether')
    abi = compile()[0]['abi']

    contract = web3.eth.contract(abi=abi,
address=database['mgmtContract'])

    try:
        tx_hash = contract.functions.setFee(fee).transact({
            'from': account['account']
        })

        wait_tx_receipt(web3, tx_hash)
        print('Updated successfully')
    except ValueError:
        print('No permissions to change the service fee')

```

Задача Utils:

```

from solc import compile_files
import time

def compile():
    compiled = compile_files([
        './contracts/ManagementContractInterface.sol',
        './contracts/BatteryManagementInterface.sol',
        './contracts/ServiceProviderWallet.sol',
        './contracts/ERC20TokenInterface.sol'
    ])

    compiled_mgmt =
compiled['./contracts/ManagementContractInterface.sol:ManagementContractIn
terface']
    compiled_wallet =
compiled['./contracts/ServiceProviderWallet.sol:ServiceProviderWallet']
    compiled_bat =
compiled['./contracts/BatteryManagementInterface.sol:BatteryManagementInte
rface']
    compiled_erc =
compiled['./contracts/ERC20TokenInterface.sol:ERC20TokenInterface']

    return compiled_mgmt, compiled_wallet, compiled_bat, compiled_erc

def wait_tx_receipt(web3, tx_hash, sleep_interval=0.5):
    while True:
        tx_receipt = web3.eth.getTransactionReceipt(tx_hash)
        if tx_receipt:
            return tx_receipt
        time.sleep(sleep_interval)

def split(arr, size):
    arrs = []

```

```

while len(arr) > size:
    pice = arr[:size]
    arrs.append(pice)
    arr = arr[size:]
arrs.append(arr)
return arrs

```

Задача Vendor:

```

from utils import compile, wait_tx_receipt, split
from web3 import Web3
import os
import sys
import time
from db import write_account, write_database, read_account, read_database,
create_firmware
from web3.middleware import geth_poa_middleware
import warnings

warnings.simplefilter("ignore", category=DeprecationWarning)

web3 = Web3()
web3.middleware_stack.inject(geth_poa_middleware, layer=0)

account = read_account()
database = read_database()

if account.get('account'):
    web3.personal.unlockAccount(account['account'],
account['password'])

command = sys.argv[1]

# US-005
if command == '--new':
    res = web3.personal.newAccount(sys.argv[2])
    print(res)
    write_account({'account': res, 'password': sys.argv[2]})

# US-006
elif command == '--regfee':
    abi = compile()[0]['abi']

    contract = web3.eth.contract(abi=abi,
address=database['mgmtContract'])
    fee =
web3.fromWei(contract.functions.registrationDeposit().call(), "ether")

    print(f'Vendor registration fee: {float("{:.6f}".format(fee))}')

# US-007 and US-008
elif command == '--reg':
    abi = compile()[0]['abi']
    value = web3.toWei(float(sys.argv[3]), 'ether')

    contract = web3.eth.contract(abi=abi,
address=database['mgmtContract'])
    fee = contract.functions.registrationDeposit().call()
    name = bytes(sys.argv[2], encoding='utf-8')

```

```

    if value < fee:
        print('Failed. Payment is low.')
    elif web3.eth.getBalance(account['account']) < value:
        print('Failed. No enough funds to deposit.')
    elif contract.functions.isVendorNameTaken(name).call():
        print('Failed. The vendor name is not unique.')
    elif
contract.functions.vendorAddressToExists(account['account']).call():
        print('Failed. The vendor address already used.')
    else:
        tx_hash = contract.functions.registerVendor(name).transact(
            {'from': account['account'], 'value': value})
        tx_receipt = wait_tx_receipt(web3, tx_hash)

        filter = web3.eth.filter({
            'address': database['mgmtContract'],
            'fromBlock': tx_receipt.blockNumber,
            'toBlock': tx_receipt.blockNumber,
            'topics': [

web3.sha3(text='Vendor(address,bytes4)').hex()
            ]
        })
        found_log = web3.eth.getFilterLogs(filter.filter_id)[0]

        print('Success.')
        id = found_log['data'][66:74]
        print(f'Vendor ID: {id}')

# US-009
elif command == '--batfee':
    abi = compile()[0]['abi']

    contract = web3.eth.contract(abi=abi,
address=database['mgmtContract'])
    fee = contract.functions.batteryFee().call({"from":
account["account"]})

    print(f'Production fee per one battery: {web3.fromWei(fee,
"ether")}')

# US-010, ...
elif command == '--bat':
    num = int(sys.argv[2])
    if len(sys.argv) == 3:
        deposit = 0
    else:
        deposit = web3.toWei(float(sys.argv[3]), "ether")
    abi = compile()[0]['abi']

    contract = web3.eth.contract(abi=abi,
address=database['mgmtContract'])

    ids = []
    private_keys = {}
    for i in range(num):
        acc = web3.eth.account.create()
        ids += [bytes.fromhex(acc.address[2:])]
        private_keys[ids[i].hex()] = acc.privateKey
    max_gas = web3.eth.getBlock('latest')['gasLimit']
    num_for_tx = max_gas // 70000

```

```

ids_split = split(ids, num_for_tx)
tx_list = []
logs = []
try:
    for ids_lst in ids_split:
        tx_hash =
contract.functions.registerBatteries(ids_lst).transact(
    {'from': account['account'], 'value':
deposit}))
        tx_receipt = wait_tx_receipt(web3, tx_hash)

        vendor_id =
contract.functions.vendorId(account['account']).call().hex()
        filter = web3.eth.filter({
            'address': database['mgmtContract'],
            'fromBlock': tx_receipt.blockNumber,
            'toBlock': tx_receipt.blockNumber,
            'topics': [

web3.sha3(text='NewBattery(bytes4,bytes20)').hex(),

f'0x{vendor_id}0000000000000000000000000000000000000000000000000000000000000000
00000'

            ]
        })
        found_logs =
web3.eth.getFilterLogs(filter.filter_id)
        logs += found_logs
        deposit = 0

except ValueError:
    print('Failed. No enough funds to register object.')
else:
    for log in logs:
        addr = log["data"][2:42]
        print(f'Created battery with ID: {addr}')
        create_firmware(addr, private_keys[addr])

# US-012
elif command == '--deposit':
    abi = compile()[0]['abi']

    contract = web3.eth.contract(abi=abi,
address=database['mgmtContract'])

    if
contract.functions.vendorAddressToExists(account['account']).call():
        print(f'Deposit:
{web3.fromWei(contract.functions.vendorDeposit(account["account"]).call(),
"ether")}')
    else:
        print(f'Vendor account is not registered.')

elif command == '--owner':
    mgmt, _, battery_mgmt = compile()[3]

    contract = web3.eth.contract(abi=mgmt['abi'],
address=database['mgmtContract'])
    battery_contract = web3.eth.contract(abi=battery_mgmt['abi'],
address=contract.functions.batteryManagement().call())

```

```

try:
    if '.' in sys.argv[2] or len(sys.argv[2]) != 40:
        lst = list(map(lambda x : bytes.fromhex(x[2:42]),
list(open(sys.argv[2])))
        tx_hash =
battery_contract.functions.transfer(sys.argv[3], lst).transact(
        {'from': account['account']})
    else:
        tx_hash =
battery_contract.functions.transfer(sys.argv[3],
bytes.fromhex(sys.argv[2])).transact(
        {'from': account['account']})
        wait_tx_receipt(web3, tx_hash)
        print('Success')
except ValueError:
    print('Failed. Not allowed to change ownership.')

```

Программы для решения полной финальной задачи нет.