Team Wallet Task.

Solidity SmartContracts.

Team 6.
Vlad Tagunkov

Task from site - dapp-world.com

dapp-world.com/problems

App World Beta | SmartBooks | Search souls | Upgrade

- My Progress
- Leaderboard

CATALOG

- Courses
- Quizzes
- Problems
- Contests

Problems

Apply your learnings!

Developer's Arena — Competitive Programming — for blockchain. Solve problem statements curated for blockchain, compete with others and sharpen your skills.

Sr. Problem	Level	Max Score	Solve
1. Storage Smart Contract	Easy	280	Solve →
2. Calculate Factorial	Easy	280	Solve →
3. Compare Strings	Easy	280	Solve →
4. Owner Smart Contract	Easy	280	Solve →
5. Find Greatest	Easy	280	Solve →
6. Check Even Number	Easy	280	Solve →
7. Yeild Farming contest-14	Hard	460	Solve →
8. DAO Voting contest-14	Medium	380	Solve →
9. Payment Channel contest-14	Easy	340	Solve →
10. Crowdfunding - Hard contest-13	Hard	580	Solve →
11. Crowdfunding - Easy contest-13	Medium	420	Solve →
12. Simple Operations contest-13	Easy	370	Solve →
13. Decentralised Roulette contest-12	Hard	600	Solve →
14. Cryptocurrency Round Trip contest-12	Medium	470	Solve →
15. Vowel Removal contest-12	Easy	400	Solve →
16. Auction contest-11	Hard	640	Solve →
17. Decimal to Binary - Hard contest-11	Medium	420	Solve →

Task - Team Wallet - Hard.

30. Fibonacci Sequence	contest-7	Easy	340	Solve →
31. Blockchain Gaming Ecosystem	contest-6	Hard	1120	Solve →
32. Bus Ticket	contest-6	Medium	420	Solve →
33. Pythagoras Theorem	contest-6	Easy	330	Solve →
34. Team Wallet - Hard	contest-5	Hard	1620	Solve →
35. Team Wallet - Easy	contest-5	Hard	1320	Solve →
36. Triangle Inequality	contest-5	Easy	820	Solve →

Task - Team Wallet - Hard.

30. Fibonacci Sequence	contest-7	Easy	340	Solve →
31. Blockchain Gaming Ecosystem	contest-6	Hard	1120	Solve →
32. Bus Ticket	contest-6	Medium	420	Solve →
33. Pythagoras Theorem	contest-6	Easy	330	Solve →
34. Team Wallet - Hard	contest-5	Hard	1620	Solve →
35. Team Wallet - Easy	contest-5	Hard	1320	Solve →
36. Triangle Inequality	contest-5	Easy	820	Solve →

Task and Interface Overview.

dapp-world.com/problem/team-wallet-hard/problem

App World Beta

Problems

Problem | Test | My Submissions | Discussions | Leaderboard

Team Wallet - Hard

This is a hard version of this problem. The only difference between the easy and hard is, an additional function - *transactionStats* , needs to be implemented in the hard version.

'Pied Piper' has emerged as the winning team in the DApp World Arena, the annual web3 multi-player gaming fest. The winning prize of the tournament is a huge amount of money, which will be given in form of credits. For this, DApp World Arena will be sharing the credits in form of a smart contract based shared wallet.

How will the wallet be shared?

- This wallet will be in form of a smart contract.
- The smart contract will be deployed by the DApp World Arena Judge.
- After deploying the smartcontract, the deployer will initialize the smart contract with all the addresses of the members of Pied Piper and the winning prize credit amount by running a one time accessible function of smartcontract.

Now, the smart contract will be ready for the winning team members to use.

How will the winning team use this smart contract?

- After the smart contract is completely set to use for the winning team members, the members of the team can create transaction requests inside the smart contract to spend the credits from the smart contract.
- There is no limit on the number of transaction requests that can be created in the smart contract.
- Any transaction request needs approval from at least 70% of the team members to be completed.
- Once a transaction request gets enough approvals, the request will be completed successfully and the credits from the wallet will be spent.
- A transaction request can also be rejected by the team members. If a transaction request gets rejections by more than 30% of the team members, then the transaction request will be marked as failed, and the credits will not be spent for that transaction.
- Any transaction request which has a spending amount greater than the current available credits in the wallet must automatically fail.

Impressed by your smart contract programming skills, DApp World Arena has selected you to create and give them the required smart contract. Given below are the required public functions which the smart contract must contain :

Input:

setWallet(address[] members, uint credits) public: This function is accessible only to the deployer of the smartcontract. The members array will have the addresses of all the members of the winning team. 'members' must contain at least one member address. The credits must be strictly greater than 0. The deployer of the smart contract cannot be a member of the team. This function must only execute once, and should not be callable once successfully executed.

spend(uint amount) public: This function will be accessible only to the winning team members. Using this function, a team member can record a transaction request to the smart contract. The amount should be strictly greater than 0. By default, an approval will be recorded for the transaction from the member who is sending transaction request.
Note : Any spend request would be recorded in the smart contract irrespective of the amount, even if the amount exceeds the available credits.

approve(uint n) public: This function will be accessible only to the winning team members. Using this function, a team member can record an approval for nth transaction request sent to the smart contract. This function will revert if the team member has already approved or rejected the nth transaction request.

reject(uint n) public: This function will be accessible only to the winning team members. Using this function, a team member can record a rejection for nth transaction request sent to the smart contract. This function will revert if the team member has already approved or rejected the nth transaction request.

Output:

credits() returns (uint): This function will be accessible only to the winning team members. The output must be the current available credits in the wallet.

viewTransaction(uint n) returns (uint amount,string status):This function will be accessible only to the winning team members. 'amount' is the spent amount requested for the nth transaction request. 'status ' must be :

- "pending" : if the transaction request is pending,
- "debited" : if the transaction request has been executed and the credits have been spent,
- "failed" : if the transaction request has failed.

transactionStats() returns (uint debitedCount,uint pendingCount,uint failedCount):This function will be accessible only to the winning team members. Three output values must be as follows:

- 'debitedCount' : number of transaction requests which have been executed successfully.
- 'pendingCount' : number of transaction requests which are pending.
- 'failed' : number of transaction requests which have failed.

Example 1

Input/Output	Function	Sender address	Parameter	Returns
Input	setWallet()	Owner	([<Address 1>,<Address 2>], 1000)	
Input	spend()	Address 1	(100)	
Output	viewTransaction()	Address 1	(1)	[100, "pending"]
Input	approve()	Address 2	(1)	
Output	viewTransaction()	Address 2	(1)	[100, "debited"]

Code Editor

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

contract TeamWallet {
    address owner;
    uint credit_pool;
    uint tx_counter=1;

    bool set_flag;
    bool member_flag;

    address[] winning_team_array;
    Request[] request_array;
    address[] spend_vote;
    //mapping(uint=>string) txn_status;
    mapping(string=>uint) txn_stats;

    struct Request{
        address requester;
        uint request_amount;
        uint tx_number;
        uint vote_counter;
        uint reject_counter;
        string status;
        mapping(address=>bool) vote_owner;

    }
    struct Approuve{
        address aprouver;
        bool voted;
    }

    constructor(){
        owner=msg.sender;
    }

    //For setting up the wallet
    function setWallet(address[] memory members, uint256 credtis) public {
        require(msg.sender==owner,"Not owner");
        require(members.length>0,"No any members");
        require(credtis>0,"Not enough credit");
        require(set_flag==false,"Already set");
        for(uint i=0; i<members.length;i++){
            if(members[i]==owner){
                revert("Owner can't be team member");
            }
        }
        winning_team_array=members;
        credit_pool=credtis;
        set_flag=true;
    }

    //For spending amount from the wallet
    function spend(uint256 amount) public {
        require(amount>0,"Not enough credit");
        member_flag=false;
        for(uint i=0;i<winning_team_array.length;i++){
            if(winning_team_array[i]==msg.sender){
                member_flag=true;
```

Compile | Submit

Description of the task.

'Pied Piper' has emerged as the winning team in the DApp World Arena, the annual web3 multi-player gaming fest. The winning prize of the tournament is a huge amount of money, which will be given in form of credits. For this, DApp World Arena will be sharing the credits in form of a smart contract based shared wallet.

How will the wallet be shared?

- This wallet will be in form of a smart contract.
- The smart contract will be deployed by the DApp World Arena Judge.
- After deploying the smartcontract, the deployer will initialize the smart contract with all the addresses of the members of Pied Piper and the winning prize credit amount by running a one time accessible function of smartcontract.

Now, the smart contract will be ready for the winning team members to use.

How will the winning team use this smart contract?

- After the smart contract is completely set to use for the winning team members, the members of the team can create transaction requests inside the smart contract to spend the credits from the smart contract.
- There is no limit on the number of transaction requests that can be created in the smart contract.
- Any transaction request needs approval from at least 70% of the team members to be completed.
- Once a transaction request gets enough approvals, the request will be completed successfully and the credits from the wallet will be spent.
- A transaction request can also be rejected by the team members. If a transaction request gets rejections by more than 30% of the team members, then the transaction request will be marked as failed, and the credits will not be spent for that transaction.
- Any transaction request which has a spending amount greater than the current available credits in the wallet must automatically fail.

Impressed by your smart contract programming skills, DApp World Arena has selected you to create and give them the required smart contract. Given below are the required public functions which the smart contract must contain :

Input and Output Description.

Input:

setWallet(address[] members, uint credits) public: This function is accessible only to the deployer of the smartcontract. The members array will have the addresses of all the members of the winning team. 'members' must contain at least one member address. The credits must be strictly greater than 0. The deployer of the smart contract cannot be a member of the team. This function must only execute once, and should not be callable once successfully executed.

spend(uint amount) public: This function will be accessible only to the winning team members. Using this function, a team member can record a transaction request to the smart contract. The amount should be strictly greater than 0. By default, an approval will be recorded for the transaction from the member who is sending transaction request.
Note : Any spend request would be recorded in the smart contract irrespective of the amount, even if the amount exceeds the available credits.

approve(uint n) public: This function will be accessible only to the winning team members. Using this function, a team member can record an approval for nth transaction request sent to the smart contract. This function will revert if the team member has already approved or rejected the nth transaction request.

reject(uint n) public: This function will be accessible only to the winning team members. Using this function, a team member can record a rejection for nth transaction request sent to the smart contract. This function will revert if the team member has already approved or rejected the nth transaction request.

Output:

credits() returns (uint): This function will be accessible only to the winning team members. The output must be the current available credits in the wallet.

viewTransaction(uint n) returns (uint amount,string status):This function will be accessible only to the winning team members. 'amount' is the spent amount requested for the nth transaction request. 'status ' must be :

- "pending" : if the transaction request is pending,
- "debited" : if the transaction request has been executed and the credits have been spent,
- "failed" : if the transaction request has failed.

transactionStats() returns (uint debitedCount,uint pendingCount,uint failedCount):This function will be accessible only to the winning team members. Three output values must be as follows:

- 'debitedCount' : number of transaction requests which have been executed successfully.
- 'pendingCount' : number of transaction requests which are pending.
- 'failed' : number of transaction requests which have failed.

Test for the Task completion.

dapp-world.com/problemreport/42605

App World Beta | Courses | Quizzes | Problems | Contests | SmartBooks | Search

Team Wallet - Hard (#42605)

Level : Hard Timestamp : 19 Sep 2024 18:34:52

Score : 1620/1620 Deploy Gas : 2537064 Transaction Gas : 16734112

Testcases — 15 Passed 0 Failed

- Testcase 1 ✓
- Testcase 2 ✓
- Testcase 3 ✓
- Testcase 4 ✓
- Testcase 5 ✓
- Testcase 6 ✓
- Testcase 7 ✓
- Testcase 8 ✓
- Testcase 9 ✓
- Testcase 10 ✓
- Testcase 11 ✓
- Testcase 12 ✓
- Testcase 13 ✓
- Testcase 14 ✓
- Testcase 15 ✓

Result

Passed

Message

Got expected output for all transactions

Testcase value

Input/Output	Function	Sender address	Parameter	Returns
Input	setWallet()	Owner	([<Address 1>, <Address 2>, <Address 3>], 1000)	
Output	transactionStats()	Address 1	()	[0,0,0]
Input	spend()	Address 1	(100)	
Output	transactionStats()	Address 2	()	[0,1,0]
Input	approve()	Address 2	(1)	
Output	transactionStats()	Address 1	()	[0,1,0]
Input	spend()	Address 2	(800)	
Output	transactionStats()	Address 2	()	[0,2,0]
Input	spend()	Address 1	(50)	
Output	transactionStats()	Address 2	()	[0,3,0]
Input	reject()	Address 1	(2)	
Output	transactionStats()	Address 1	()	[0,2,1]
Input	reject()	Address 2	(3)	
Output	transactionStats()	Address 2	()	[0,1,2]
Input	spend()	Address 2	(50)	
Output	transactionStats()	Address 1	()	[0,2,2]
Input	approve()	Address 1	(4)	
Output	transactionStats()	Address 2	()	[0,2,2]
Input	approve()	Address 3	(1)	
Output	transactionStats()	Address 2	()	[1,1,2]
Input	approve()	Address 3	(4)	
Output	transactionStats()	Address 2	()	[2,0,2]

Gas Used

1583826

AI Assistant

Submission Example.

App World Beta

Problem | Test | My Submissions | Discussions | Leaderboard

My Submissions

#ID	Result	Score	Timestamp
42605	Passed	1620	19 Sep 2024 18:34:52
39496	Passed	1620	19 Sep 2024 16:22:11
39485	Passed	1620	19 Sep 2024 16:19:20
39484	14/15 Passed	1520	19 Oct 2024 16:18:52
39482	Passed	1620	19 Sep 2024 16:18:18
39447	14/15 Passed	1520	19 Sep 2024 16:12:04

Leaderboard by Total Exp.

dapp-world.com/leaderboard

App World Beta | SmartBooks | Search souls | Upgrade

- My Progress
- Leaderboard

CATALOG

- Courses
- Quizzes
- Problems
- Contests

Leaderboard

See how you stack up against other developers!

Sort By: Total Exp | Page Size: 25

# Rank	Username	Rating	Total Exp
4	TVID4d	1474	18400
1	Of3lija	2369	28550
2	bepossible	2558	28250
3	bg5fxp_JF	1896	19850
4	TVID4d	1474	18400
5	Shoydon	1834	16250
6	theblockmatrix	1691	15700
7	Sheetal3d8	1368	15150
8	ad25	1880	15000
9	The3l6	1049	14650
10	RavikantShinde	0	14550

Leaderboard by Total Exp.

dapp-world.com/leaderboard

App World Beta | SmartBooks | Search souls | Upgrade

- My Progress
- Leaderboard

CATALOG

- Courses
- Quizzes
- Problems
- Contests

Leaderboard

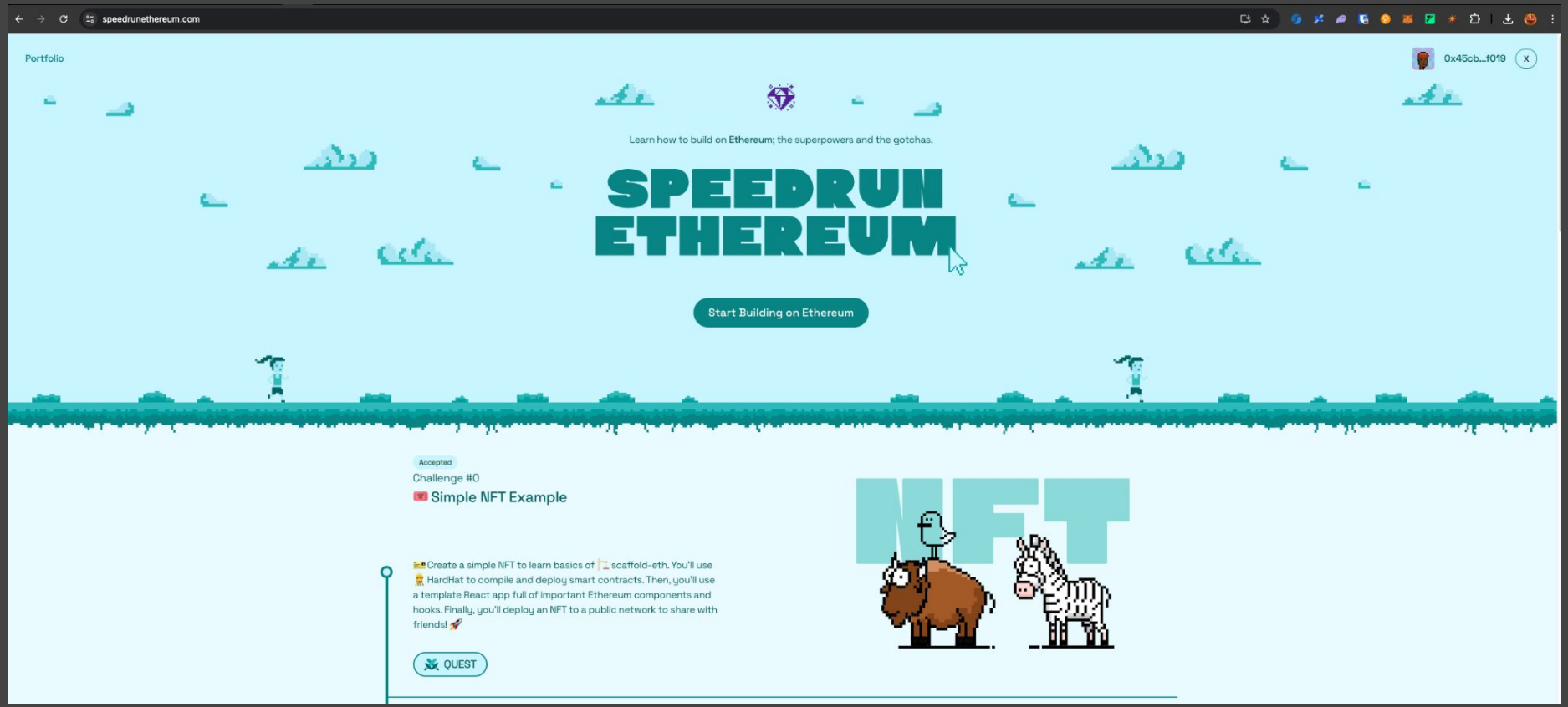
See how you stack up against other developers!

Sort By: Total Exp | Page Size: 25

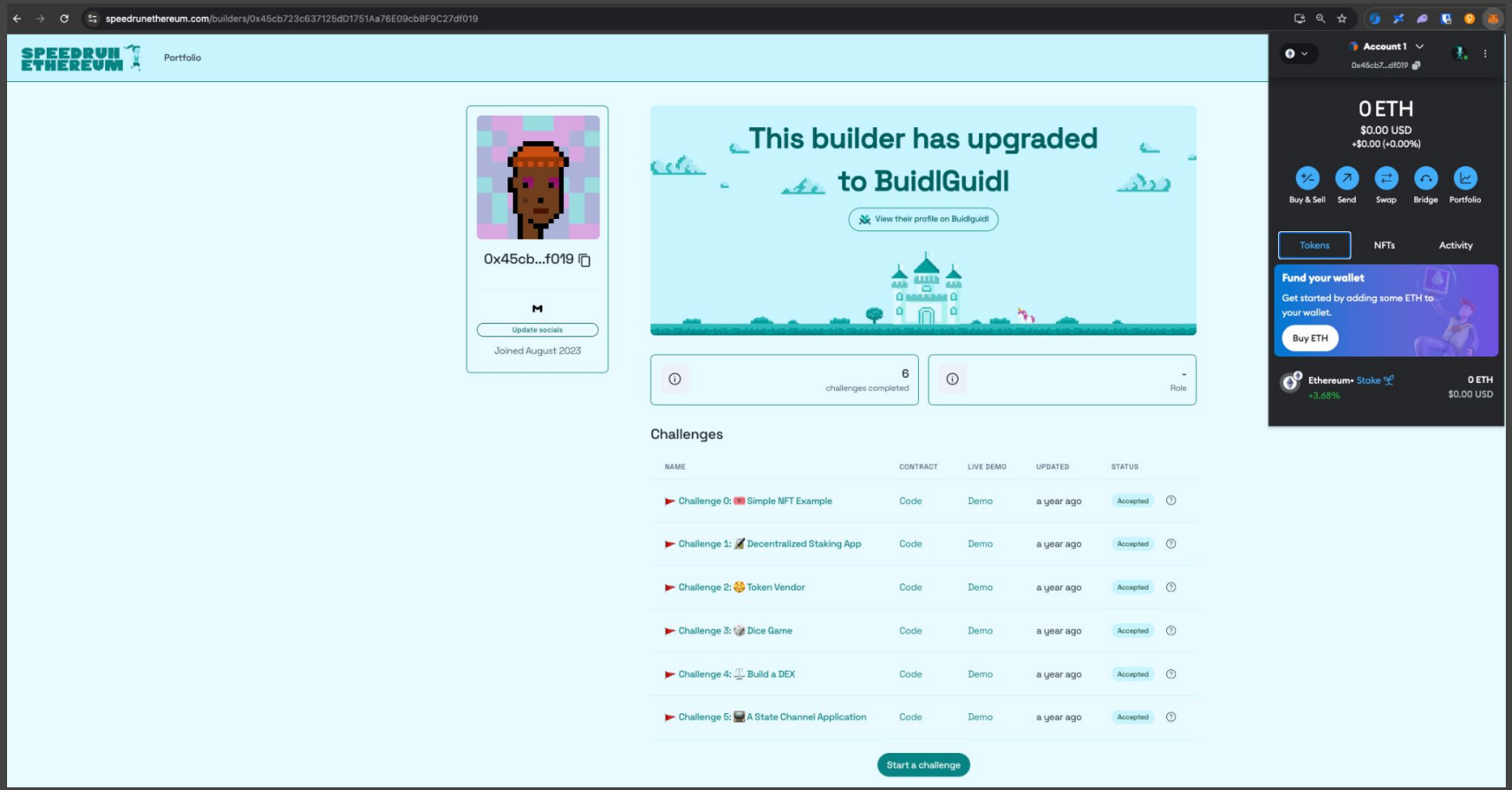
# Rank	Username	Rating	Total Exp
4	TVID4d	1474	18400
1	Of3lija	2369	28550
2	bepossible	2558	28250
3	bg5fxp_JF	1896	19850
4	TVID4d	1474	18400
5	Shoydon	1834	16250
6	theblockmatrix	1691	15700
7	Sheetal3d8	1368	15150
8	ad25	1880	15000
9	The3l6	1049	14650
10	RavikantShinde	0	14550


JUST ONE MORE THING.

Task from SpeedRunEthereum.com



Great task with validator.



Thank you.