**Solidity developer tech assignment**

Problem:   
Your task is to create a staking contract.  
Users stake different quantities of ERC-20 token named “TKN”. Assume that an external caller would periodically transfer reward TKNs to a staking smart contract (no need to implement this logic). Rewards are proportionally distributed based on staked TKN.  
Contract caller can:  
- Stake  
- **Unstake (would be a plus if caller can unstake part of stake)**  
- See how many tokens each user can unstake  
Main info:  
- Use Solidity as a main language (version higher than 0.4.0)  
- Cover contract with unit tests. You can use Truffle or Hardhat

**My Word:**

I think this assignment is really worth for getting notice the ability of solidity developer.

To be honest, I have read many similar projects and papers to find the best solution to be passes in this step.

Actually, during that period, I noticed that the essay ***[Scalable Reward Distribution on the Ethereum Blockchain](https://uploads-ssl.webflow.com/5ad71ffeb79acc67c8bcdaba/5ad8d1193a40977462982470_scalable-reward-distribution-paper.pdf)*** [by Batog et al](https://uploads-ssl.webflow.com/5ad71ffeb79acc67c8bcdaba/5ad8d1193a40977462982470_scalable-reward-distribution-paper.pdf)is not suit one to meet all requirement from this assignment.

- Main issue is that

The paper by Bogdan et al. assumes that stake size for each user doesn’t change once it is set(stakeholder deposit tokens or ether into staking pool). According to this assumption, stakeholder should wait to collect rewards and then can unstake his assets including rewards from staking pool.

However, it’s model can not accepted in real world, actually.

That’s why there’s requirement to change staked size anytime and unstake some of them. Right?

Yes, I strongly agree with this requirement and I reached out the fact that I have to modify the algorithm in this essay a little bit.

That’s why I named staking contract as **StakeV2**

- What I did

The total amount of reward of stakeholder is

This can be rewritten like following:

※ according to this equation:

I proved this by myself and if you mind, it’s okay to skip following part

Substitue on the LHS of above equation

Here:

Also according to

The **equation \*** should be

Or

Then, let’s define

We can rewrite

If we define

Then we get

That’s what I modified to implement the requirement from this assignment.

In my contract, there are 3 main variables (stake, reward\_per\_token, reward\_tally) and 3 main functions(deposit, distribute, withdraw).

Modification actions should be taken like following:

|  |  |
| --- | --- |
| **Variable** | **Function** |
| stake | deposit, withdraw\_stake |
| reward\_per\_token | distribute |
| reword\_tally | deposit, withdraw |

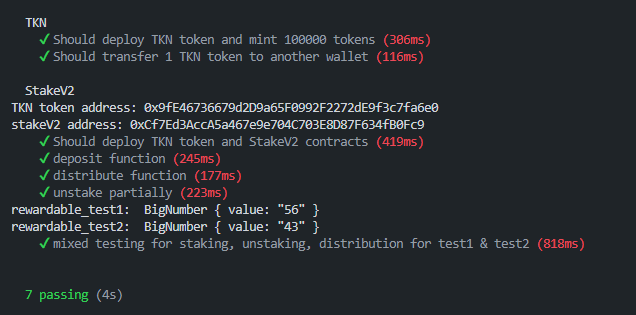
**-Unit-Testing**

1. ERC20 Token Test

Token delopy and transferring operation are tested

1. Staking Operation Test

* Tested Staking & ERC20Token contracts deploying and transfer functionality of TKN tokens in Staking contract
* Deposit functionality for test1 account.
* Distribution functionality for test1 account.
* Unstaking(partially) functionality for test1 account.(also should works fine for complete unstaking)
* Mixed Testing with test1 & test2 account for all kinds of functionality above(including showing unstakable assets)



Thank you very much