Farming Contract – COLUMBUS 5

Overview:

LoopSwap comes with the feature of Staking LP tokens and earning the reward in the form of various reward tokens.
User needs some "LP Tokens" to enter into a Farm with. Farms can only accept their own exact LP Token. For example, the LOOP-UST Farm will only accept LOOP-UST LP Tokens. To get the exact LP Token, user will need to provide liquidity for that trading pair.

Main functionalities are given below:

- Users can stake the LP tokens.
- Admin decides the Number of total reward tokens to be distributed to the pool as a reward.
- The distribute function is called after which reward tokens will be distributed.
- Proportionate to user's staked LP tokens as compared to the total USD value of the reward pool, user share in pool reward will be calculated.
- User can then unstake and claim his reward.

Methodology:

The contract is primarily based on creating a "LoopFarm" token for each pool. When user stakes his LP tokens in a particular pool, the contract mints him "LoopFarm" (uLF) tokens. These uLF tokens are minted to the user based on the ratio of the LP staked to the total USD value of the pool. In this way, a user who stakes LP tokens earlier than others gets more uLF tokens in comparison to the user who stakes equal number of LP tokens later because the USD value of the reward pool has increased due to subsequent distributions of the reward tokens by the admin.
The contract keeps track of the LP staked by a user through the hashmap `LP_PROVIDED` so that during unstaking and claiming, user can get his due LP tokens returned to him. Additionally, the functions of unstaking and claiming have been fused together so that those users who keep their LP tokens staked are benefitted. For instance, if a user decides to "unstake and claim", his LP tokens will be unstaked and rewards will be claimed but when he restakes the LP tokens, he will get lesser uLF tokens because the pool value has increased during this time and the period when he staked for the first time. So in this way, a user who keeps "unstaking and claiming" will be at a relative disadvantage than a user who keeps his LP tokens staked in the pool.

Functions

`pub fn instantiate():`

This function will be used to Initialize the smart contract. It needs no parameter to initialize.

`pub fn execute update config():`

This function is called by the admin when admin wants to change ownership of the contract.

`pub fn execute add stakeable token():`

This function will allow admin to add the assets to be staked. In the start it performs the authorization check through `state.owner` and then checks if token already exists in the stakeable token list. Finally, it allows the admin to add new stakeable tokens and then mints a new FLP token corresponding to that particular pool based on what user share in reward will be calculated.

`pub fn execute add distribution token():`

This function will allow admin to add the assets to be distributed as a reward e.g. LOOP tokens etc. to the users staking LP tokens. In the start it performs the authorization check through `state.owner` and then checks if token already exists in the distribution list. Finally, it allows the admin to add new distribution assets.

`pub fn execute stake():`

This function will be called by the contract to stake user LP tokens when user sends asset to the contract in accordance with the receive interface of the contract.
It will take only one parameter i.e. `asset` - the details of LP tokens user wants to stake. First it will check if the asset user wants to stake is among the stackable assets. It keeps a record of the staked value through `LP-PROVIDED` and also maintains the record of `TOTAL_STAKED`. Based on the USD value of the rewards in pools, it mints FLP tokens to the user respective of the share in the pool. When rewards are distributed by the admin, subsequently, new user will get lesser share when he stakes the same amount of LP tokens as the previous user who staked before him. It also keeps tracks of the `REWARD_TOKEN_ISSUED` to the user through the hashmap.

pub fn execute_update_reward():

This function will update daily reward for a pool.
It will take two parameters

- array of reward tokens
- pool token address

First of all, it will check whether the pool exists or not. Then it will check for each reward admin wants to update whether it exists in the distributable tokens or not. Then it will update the reward.

pub fn execute distribute():

This will be called by the admin to distribute daily rewards as per data in `pub fn execute_add_distribution_token()`

pub fn execute_unstake and claim():

This function will be called by the contract to unstake the LP tokens of the user and claim his rewards when the contract receives uLF tokens from the user with the `unstake_and_claim` message in accordance with the receiver interface of the contract.
It will take only one parameter i.e. `asset` – the address and amount of uLF tokens user wants to send to the contract to unstake proportionate LP tokens. The contract calculates the equivalent FLP tokens that user wants to unstake. The contract receives uLF tokens, transfers LP tokens back to user, burns the uLF tokens and then transfers the rewards that the user had accumulated in the particular pool.
The function updates `LP_PROVIDED`, `TOTAL STAKED`, `REWARD_TOKEN_ISSUED`, and `TOTAL REWARD IN POOL`.
Note: If user will try to unstake the amount greater than he staked, the initial "if" condition will avoid any transaction and return an error.

Queries

`pub fn query reward in pool():`

This query will return the total available reward of a particular distribution token in the specific pool. It requires both `pool` and `distribution token` addresses in order to load value from a hashmap.

`pub fn query_reward token to user():`

This query will return the FLP tokens minted to the user for a particular staking pool through the hashmap `REWARD_TOKEN_ISSUED`.

`pub fn query list of distributable tokens by pool ():`

This query will return all the distributable tokens (rewards) that are available for claiming in the given pool through the `fn query pool rewards`.

`pub fn query_user_reward_in_pool ():`

This query will return the user reward – all distribution tokens in a particular pool – based on user staked LP tokens.

`pub fn query_staked_by_user():`

This query will return the value of LP tokens user has staked through the hashmap `LP_PROVIDED`.

`pub fn query_total_staked():`

This query will return the total staked amount in the pool by all the users through the hashmap `TOTAL_STAKED`.

`pub fn query_apr():`

This query will calculate the APR of the given pool asset.

`pub fn query_list_of_stakeable_tokens():`

This query will return the list of all stakeable tokens (pools).

pub fn query_list_of_distributable_tokens():

This query will the list of all distributable tokens (rewards).

pub fn query_price():

This query will return the current price of the token against usd. It is used in `fn execute_stake` to evaluate the usd equivalent pool size.

pub fn query_price2():

This query will return the current price of the token against usd. It is used in `query_apr()`.

pub fn query_stakeable_info():

This is for testing only and returns the `STAKEABLE_INFOS`.