Celery (Smart Contract)

**Contract requirements**

1. Account and address can be used interchangeable
2. The contract must keep track of each account, using a key-value pair. The key is the address of the account, and the value is an “Account” object with various properties needed to keep track of each account
3. Each account can either be in a “Stake” or “Payout” stage.
4. Switching from “Payout” to “Stake” will cause a “cash out” of what was earned in the payout stage (how long it was in payout)
5. Switching from “Stake” to “Payout” will cause the current stake amount (which will be calculated from when the Stake stage started) to start to payout, and it will no longer increase in value

**User Stories:**

1. User wants to create an account
   1. An account is implicitly created when an action is taken against the contract.
   2. For a brand new contract, adding a stake amount to the contract is how you would create the account. This will begin the staking progress
   3. Note that there is no “holding” area for the tokens. When you add tokens to the contract, they are automatically bound by the rules of staking and payout.
   4. Since staking is the only way to “start” the contract, that will be the first stage for every account
2. User wants to add more tokens to stake
   1. A user can add more tokens to the contract at anytime. When this is done, the contract will switch into “Stake” stage if it hasn’t been already. If this occurs, a payout may occurs if any payout was earned while in the “payout” stage (again, there is no holding area for coins)
3. User wants to switch from staking to payout
   1. The tokens in the contract will stop staking (and calculate the new value they have per the rules for how long they were staking) and begin paying out over the specified time
   2. Again, if a user adds more tokens during this stage, it will automatically payout the earns to the user during this stage, and switch back to staking
4. User wants to switch from payout to staking
   1. This will cause the user to be paid out for what they earned in the payout stage, and the remaining tokens will start staking
5. User wants to collect their payout
   1. The account must be in the payout stage for this to work. If not, this will not payout any tokens and will NOT switch the stage to payout
   2. If in the payout stage, this will deposit all tokens earned so far since the last payout into their account
6. User wants to collect all of their tokens back at once
   1. This is the same as the above collect payout, but will take out all tokens in the contract, with a penalty instead of a time-based withdraw

**Technical Details:**

1. For each account
   1. We need to store how many tokens are in the account.
      1. This amount will be updated when a user adds more tokens to their account.
      2. It is also updated to new values as the account is transitioned from payout to stake (to subtract what will be withdrawn) and stake to payout (to add what was gained in the stake).
      3. It will also subtract if a separate payout command is executed (to subtract what was collected) or set to 0 when a payout with penalty is performed
   2. We need to store the time of the last write operation that occurred
      1. When switching from stake to payout, we need to take that time and subtract it from the current time to determine how many tokens the user earned during the stake
      2. When switching from payout to stake, we need to take that time and subtract it from the current time to determine how many tokens the user will be paid out
      3. After that operation occurs, we need to update that stored time with the current time, so the next time a write operation occurs, we can use it as a comparison
      4. A good rule of thumb: If we are changing the variable of how many tokens the user as in their account, after that operation, we should update this time field with the current time
   3. We need to store the current payout value
      1. Since collect will remove a certain amount of tokens from the contract, we need to know the baseline since the payout happens over the course of X days (and this shouldn’t change each time a collect occurs(
      2. Therefore, this value is only updated when going from stake to payout stage. In any other write transaction (or in stake stage), this value should be ignored as it means nothing
   4. We need to store the current stage of the account (Stake or Payout)
2. Each operation
   1. Increase Stake
      1. User is depositing X coins into the contract
      2. If we are in payout mode, we take the current time and stored time, check the payout earnings, pay it out back to the user, subtract from stored amount, and update the current stored time to the current time
      3. We then add the new tokens to their stored amount, and switch the stage to Stake stage
      4. Return if a withdraw occurred or not (and stage change)
      5. Throw on failure
   2. Start Stake
      1. Throw if in stake mode already
      2. If we are in payout mode, we take the current time and stored time, check the payout earnings, pay it out back to the user, subtract from stored amount, and update the current stored time to the current time
      3. We then switch the stage to stake stage
      4. Return if a withdraw occurred or not (and stage change)
      5. Throw on failure
   3. Start Payout
      1. Throw if in payout stage already
      2. If we are in stage mode, we take the current time and stored time, check the staging earnings, add it to the stored amount, and update the current stored time to the current time
      3. We then switch the stage to payout stage
      4. We then set the current payout value to the stored value, as this will be our snapshot of the total being paid out over X days
      5. Return if a stage switch occured
      6. Throw on failure
   4. Collect Payout
      1. Throw if in stake stage
      2. If we are in payout mode, we take the current time and stored time, check the current payout earnings (from the current payout value field), withdraw it to their account, subtract it from the stored amount, and update the current stored time to the current time
      3. Again, we leave the current payout value as is, since it can be used for future collect payout commands
      4. Return the amount that was collected