Take Home Assignment

For this assignment you will be creating a simplified DEX for synthetic assets. It should allow a user to deposit a stable asset as collateral to mint synthetic assets, contribute liquidity to an Automated Market Maker, and buy/sell Synthetic assets.

The program should be written using Anchor ≥ 0.25.0. You can deploy this to devnet or have it run locally.

The repository should also include a README and a script to run through and test the functionality.

Requirements:

1: To begin you'll need to provision an oracle as the source of truth. You have many options:

- Write your own smart contract or you can fork and edit another oracle program. Make sure you can edit the oracle price during runtime so that various price scenarios can be tested.
- Use an oracle on Devnet (Pyth, Switchboard, Chainlink)

2: The synthetic DEX should have the following instructions but is not limited to only them:

- **mint_synthetic_asset:** this instruction allows the user to deposit stable coin collateral and mint synthetic asset. The user should be able to choose how much they can mint but it should be restricted with a collateralization ratio such that `collateral_value / (oracle_price * minted_amount) >= 1.5`
  - You can create your own "mock" stable coin to be accepted by the program and you can make the assumption it's perfectly pegged to USD.
  - The position information such as collateral deposited and minted amount should be stored on-chain.
- **provide_liquidity_to_amm:** this instruction allows a user to contribute X amount of synthetic asset and Y amount of stable coin to an AMM pool.
  - If the pool is empty then `Y = oracle_price * X`

    - Otherwise `Y = pool_price * X`
    - There should be a mechanism in place to keep track of how much relative liquidity the user has provided. As an example if there is $100 of liquidity already in the pool and the user contributes $50 of liquidity. They now own 1/3 of the liquidity in the pool.
- **withdraw_liquidity_from_amm:** this instruction allows a user to withdraw any amount of their liquidity from the AMM. It should be using the tracking mechanism from before to figure out how much of each asset it can withdraw.
- **trade_synthetic_asset:** instruction to allow a user to buy or sell, the trade should follow the standard `x * y = k` constant product formula, assume no fees.

Note that there may be other instructions you may want to include in your smart contract.

3: Write a small testing script the runs through the instructions. Include enough tests to convince us that these instructions work, but we don't need you to test every single edge case or scenario.

You will be judged on your code quality, readability and satisfying the requirements laid out. Good luck!