

In this homework, you are expected to implement a smart contract that performs a liquidation through flash loan.

Prerequisite

Register an account [to an external site.](#) for access to an archive Ethereum node.
Install [dockerLinks to an external site.](#) on your machine or prepare the nodeJS environment.



Solidity Programming

Detailed Solidity documentation: <https://docs.soliditylang.org/en/latest/Links to an external site.>
Best website course for Solidity: <https://cryptozombies.io/en/course/Links to an external site.>

Requirements

The smart contract should allow you to perform a flash loan, a liquidation, and an asset exchange in **one** blockchain transaction.

Your contract is required to provide a unified interface **operate** to help ease marking. By calling **operate**, the flash loan, liquidation and asset exchange should be executed properly. You are allowed to "hardcode" the execution logic and parameters in the **operate** function.

Test case

You are expected to liquidate 0x59CE4a2AC5bC3f5F225439B2993b86B42f6d3e9F on Aave V2 which was liquidated at block 12489620. Check out the [original liquidation transactionLinks to an external site.](#)

Commands

To test your contract:

```
docker build -t liquidation-flash-loan .
```

```
docker run -e ALCHE_API="$YOUR ALCHEMY ETHEREUM MAINNET API" -it liquidation-flash-loan npm test
```

Grading

Your grade is determined by the profit you earn in the test case. After the program execution, you should see Profit xxx ETH at the end of a successful liquidation. If you are not using the docker environment, for successful execution you should see a profit.txt file which contains the amount of ETH that you earned after the liquidation. **If your implementation is correct, you should be receiving at least 21 ETH as the profit.** Note

that we reduce the gas fee to be zero to encourage programming complicated liquidation strategies.

程序代写代做 CS编程辅导

Submission

Your submission should contain one import statement for the `LiquidationOperator` interfaces from other contracts so that we have a unified interface for grading. There are some helpful



`LiquidationOperator.sol` file that contains at most one `console.sol`;. If you plan to include libraries or interfaces, please manually add them to your contract file so that we can grade your submission. The guidelines in the `LiquidationOperator.sol` file.

Background Information

Aave Liquidation

To trigger a liquidation on Aave V2, you need to call a public function `liquidationCall` provided by the Aave V2 smart contracts. In the function, you can specify user representing the borrowing position you would like to liquidate, `debtAsset`, the cryptocurrency you would like to repay (let's say token D), and `collateralAsset`, the collateral cryptocurrency you would like claim from the borrowing position (let's say token C). You also specify the amount of debt you want to repay, `debtToCover`.

function `liquidationCall`

`address collateralAsset,`

`address debtAsset,`

`address user,`

`uint256 debtToCover,`

`bool receiveAToken`

) external;

By calling this function, you then repay some amount of token D to Aave and in return, some token C is sent to your account.

You should make sure that the user is in a liquidatable state. Otherwise, the Aave smart contract would revert your transaction and you would pay transaction fees for an unsuccessful liquidation.

Uniswap flash loan

What if you don't have any upfront token D, in order to repay in the liquidation, you can use flash loans! A Uniswap flash loan can grant you the cryptocurrencies available in the pool without any collateral, as long as you preserve the constant K in the end of the

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

transaction. Check out the detailed code snippet ([Uniswap V2Links to an external site.](#)) in the following.

Importantly, Uniswap would attempt to call into the receiver to invoke the function `uniswapV2Call` after sending the flash loan assets. This means that you need a smart contract to act as the receiver. The smart contract should have an `uniswapV2Call` function. In your program how you use the flash loan assets in this function.

Back to liquidation, the liquidation logic in the `uniswapV2Call` function. So, when you don't have enough token D to perform a liquidation, you can request a flash loan and your program can do the liquidation after receiving token D.

What do you need to do after liquidation?

With the flash loan, you now have enough token D. You can repay the debt for the borrowing position and claim the collateral token C. Congratulation! A successful liquidation is completed, but, wait, you still need to repay the flash loan. Remember that you need to preserve the K. In the exercise, you are required to convert every earned token to ETH through e.g., Uniswap V2. This is for easing the grading.

Working files

[liquidation-flash-loan.zip](#)

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>

程序代写代做 CS编程辅导

WeChat: estutores

Assignment Project Exam Help