Time for our next, a little more advanced challenge (but don't worry, you awesome people can do it). This challenge requires you to learn about tokens.

What are they exactly?
Many dApps built on Ethereum have their own cryptocurrencies or "tokens." In order to interact with the dApps, users need to purchase the dApp's native token.

The question which arises here is, If we have Ether why do we need tokens? To answer this question read here.

One of the many Ethereum development standards focuses on token interfaces. These standards help ensure smart contracts remain composable. So, for instance, when a new project issues a token, it should remain compatible with existing decentralized exchanges.

In the upcoming challenges, you will be implementing the interfaces of different standards. But you can't do that without knowing what interfaces are! So, you can read about interfaces from here.

CHALLENGE # 01

You guys learned about tokens and their interfaces.
Time to learn about the standard now.

One of the most popular token standards is ERC20.
You can read about the standard here.

For this challenge here is the question. Write a smart contract to create your own token. But, here is the catch, you have to implement the ERC20 standard.

CHALLENGE # 02

You know what tokens are and you know what token standards are. But did you know that tokens are of two types? Learn more about fungible and non-fungible tokens here

ERC20 falls under the category of fungible tokens. If you completed the first challenge you already created, if you haven't, you should go back to challenge 1 and do that. Haven't coded a non-fungible token?. It's about time you do that.
The most popular non-fungible token is ERC721.
You can read about it from here

For this challenge, write a smart contract to create a token that implements the ERC721 standard.

CHALLENGE # 03
Did you know that a developer referred to as Dexaran pointed out a problem in ERC20 standard, “If you will send 100 ERC20 tokens to a contract that is not intended to work with ERC20 tokens, then it will not reject tokens because it can't recognize an incoming transaction. As a result, your tokens will get stuck at the contract's balance.”

He proposed a solution to this problem as well. Pretty cool right? This solution is another standard, ERC223.

ERC223 requires contracts to implement the ERC223Receiver interface in order to receive tokens. If a user tries to send ERC223 tokens to a non-receiver contract the function will throw in the same way that it would if you send ether to a contract without the called function being payable.

For this challenge, you have to implement his very standard. Create a smart contract and get started. Implement the ERC223 standard, also make a receiver contract, and implement ERC223Receiver.

CHALLENGE # 04

How awesome would it be if you can use your non-fungible tokens like fungible tokens!

We have a whole different standard for this purpose, ERC1155.
You can read about the standard here

Create a smart contract to create a token that follows the ERC1155 standard.

CHALLENGE # 05

Do you know what decentralized exchanges are?
A **decentralized exchange** is a trading pair matching system that allows people to place orders and trade cryptocurrencies without relying on an intermediary institution to manage the ledger and hold customers' funds. Instead, trades occur directly between users (peer to peer) through an automated process.
Feel free to search and learn more about it.

For this challenge, you have to create a decentralized exchange. Yup, you read it right.
The catch is that users will trade ERC721 tokens on this exchange.

*HINT: Implement the order book model in the exchange contract.

4 challenges of 2 days each and 7days for the last challenge.