Hey Guys!

Wanting to get your hands dirty on solidity? To build some awesome dapps?, You're in the right place!

Here is challenge # 1

1. Create a smart contract to store and retrieve values.
   Not that tough? Here's the catch, you must use mapping and struct.
   Are you familiar with JSON objects? If yes then you know the insides of struct, and well, it is what lies inside that really matters, and inside the struct lies attributes just like JSON.
   If you're wondering why aren't we using the plain variables, remember "United we stand, divided we fall".

   Now, multiple people having the same objects? Try using key-value pairs! How? The answer lies in mapping!

In the good old days, people used to go to libraries for that, in the modern era we use google.
Fun Fact: Google is an actual word Merriam Websters, the dictionary.
For this challenge you first need to know what libraries are, nope, not the libraries I mentioned above, I mean the libraries which consist of reusable code.

First google Ownable, Pausable, and Safe Math libraries.
You can refer to openzeppelin's Ownable, SafeMath, Pausable library as well

When a contract is ownable, only the owner or creator can call the methods which use the modifier from the ownable contract.
Modifier? What is that? The modifier is similar to the law!
Kinda like how law restricts our actions for safety purposes, modifiers do the same thing.

Still, confused? You can learn about modifiers here.
Now, challenge 2 is coming through.

2. Create a function to update a variable by adding plus one to it.
   Easy? Easy indeed! All you have to do is use the above-mentioned libraries

---

Completing challenge 3 will make you glee

3. Create a To-do list using solidity, where users can add and toggle the tasks on the list.
   Time to Reutilize mapping and structs here!
   Completing challenges 1 and 2 makes this one easy peasy.
   Add the ownable functionality. I mean you wouldn't want just anyone to add tasks to your list right?

Bonjour to challenge 4

4. Create a smart contract, send and withdraw ethers.
   All you need to know is PAYABLE.
   A function needs to be of the payable type in order to be able to receive ethers.
   What if the function isn't payable? It will sadly refuse to accept the ethers and the transaction will be reverted. Want the contract to accept Ethers no matter what?
   Fallback functions make it possible
   You can send ethers by simply exploring this awesome Article and learn about the different ways to handle this case!

Let's deep dive into challenge five.

5. Create a smart contract to store ethers in it. But you already did that in challenge 3.
   Here is what you have to do, make it so that the user can't withdraw them before a specific time.

   Each block in the blockchain includes a timestamp!
   Timestamp is specified as the number of seconds since the Unix epoch.
   Unix epoch hmmm sounds complex? But only seconds have passed since 1st January 1970.
   Solidity made it easy to access timestamps. The answer lies in block.timestamp.
   The concept of timestamp is really important for solidity and since you completed this challenge, kudos to you! You have learned about timestamps.

   You can find the current UNIX timestamp here

Time to use your logistics for challenge six.

6. For this challenge, create a smart contract to generate the hash of a pair of addresses.
   Here the catch is, the user can enter any of the two addresses as first parameter,but the hash generated should be the same.

   To generate hash, two methods are used namely, sha256 and keccak256.
   You can get an overview here.

We're in the End Game Now.

As you know anyone can send a message using a smart contract. Did you know you can sign the messages using your private key? Even if you didn't, well now you know!
For this challenge, you have to write a smart contract to sign a message. Then to verify the signature we will need another function as well!

Hint: You will have to use the Remix console and you will have to connect your MetaMask account as well.

Congratulations on learning the concept of "_____"
We would love to see your progress, To make it possible remember to

- git-push the code.
- Post about your progress on Twitter
- Use our hashtags #

Every challenge will be of two days and in the end.