INTRODUCTION

I have chosen the first proposed project: Quiz Coin. I have decided to work alone.

The main goal is to create a Quiz Coin, which quiz enthusiasts may obtain for free from a Dispenser Smart Contract. They are then offered quizzes that have either textual answers or numeric answers. To participate, the quiz enthusiast can send his/her guess for the answer together with one Quiz Coin. The first one who finds the correct answer obtains a fixed price of 50 ALGOs. When a quiz has been correctly solved, no more submissions are accepted.
There are also a few extensions, such as implementing a Smart Contract that sells the Quiz Coin for 1 ALGO. Furthermore, there is a rebate scheme such that the quiz enthusiast may buy 10 Quiz Coins for only 9 ALGOs and 50 Quiz Coins for only 40 ALGOs.
Finally, if no correct answer has been submitted after 1 week, the prize money goes back to the quiz company and no more submissions are possible.

PLANNING

Given the project description, I will list the main parts and functionalities:

- Quiz Coin token creation
- Free Quiz Coin Smart Contract Dispenser
- Smart Contract for managing questions and answers (question and answer creation, check of 1 Quiz Coin submission, check of answer correctness, prize reward emission, closing of answer submissions after correct answer or 1 week, give prize money back to Quiz Company if no correct answer after 1 week)
- Smart Contract for selling Quiz Coins with the rebate scheme (not completely working)

This is the order in which I will implement those features, starting from the token creation. Then, I decided to use Smart Signatures for the Quiz Coin Dispenser, management of question and answer and Quiz Coins selling. This is because I don't necessarily need the additional features and complexity of proper Smart Contracts.
The implementation will be more detailed alongside with the code, together with all the instructions for replicating the execution.

USAGE:
Simply import all the submitted files by keeping them at the same level (no subfolders). Further instructions are provided in the notebooks. The first one contains both the creation and usage of the smart contracts.

SMART CONTRACT ADDRESSES:

Dispenser:
YVGRKP255O2MRGK4P6UVTQU2W4DNZSOMI4PXPLEQSN5WV7GJEIVDHE4QGQ

Country quiz:
VRBJLYDADJWW6NF7NEDTG6343LGUYZFPSCGVL374ZXFZEYHGDDPISQXI7M

Vending:
2OD3S53WDKSYJIFPDHC5SJIPZBDYJ2BS4UJJP7QYSS3OGPPVRGPO5DL52I

CREDENTIALS:
Create a credential file with the following format:

```
{
    "algod_test": "https://testnet-algorand.api.purestake.io/ps2",
    "algod_main": "https://mainnet-algorand.api.purestake.io/ps2",
    "index_test": "CREDENTIALS",
    "index_main": "CREDENTIALS",
    "purestake_token": {
        "X-Api-key": "CREDENTIALS"
    },
    "pinata_jwt": "CREDENTIALS",
    "pinata_secret": "CREDENTIALS",
    "MyAlgo": {
        "public": "CREDENTIALS",
        "private": "CREDENTIALS+PtgVaMn34bCrOVLSGPWPbXLMEYiCA==",
        "mnemonic": "CREDENTIALS"
    },
    "Alice": {
        "public": "CREDENTIALS",
        "private": "CREDENTIALS",
        "mnemonic": "CREDENTIALS"
    },
    "Bob": {
        "public": "CREDENTIALS",
        "private": "CREDENTIALS==",
        "mnemonic": "CREDENTIALS"
    },
}
```

MyAlgo and Bob need to be funded with at least 15 ALGOs, for doing some testing.

FUTURE DEVELOPMENTS:

- Web interface for both the Dispenser and Vending contracts
- Web interface for checking open quizzes and sending answers
- More checks on the contracts, such as avoiding that the Quiz Company can submit answers
- Improved management of errors and transaction failures, for example when submitting wrong answer or requesting incorrect quantities of Quiz Coins
- Fixing of the vending rebate scheme