Assignment 3

Due: 11:55 pm 27 Feb 2021
Total Mark: 100 (17% of Final Mark)

General Instructions: Please read the following instructions carefully.

Q1. Create a simple blockchain system for playing a tic-tac-toe game between two parties, Alice and Bob.

Both players have an initial block to start their play. Let Alice always start first for simplicity. The initial block, *Block0.json,* for both Alice and Bob is set as follow:

```
{
        "TxID": 0,
        "Hash": "This is the genesis block.",
        "Nonce": 0,
        "Transaction": []
}
```

Figure 1. Block0.json

At each turn, a player (e.g., Alice) who takes the turn creates a new block in JSON format and send it to the other player (e.g., Bob) who plays tic-tac-toe together. Each block is formed as follows:

```
{
        "TxID": <TxID>,
        "Hash": "<Hash> ",
        "Nonce": <Nonce>,
        "Transaction": <Transaction>
}
```

The entities in the above block are computed in a way described as follows:

1. <TxID> is an incremental counter, which increased by 1 on each turn.

   $$\langle TxID \rangle \text{ of a new block} = \langle TxID \rangle \text{ of the last block} + 1$$

2. "Hash" is the output of the SHA256 hash with the last block and "Nonce". The player increases the value of "Nonce" from 0 by 1 until the value of the hash output is less than 2^{248}.

   $$\langle Hash \rangle = SHA256(\langle \text{The last block} \rangle || \langle Nonce \rangle) < 2^{248}$$

   HINT: You may need to convert *<The last block>* and *<Nonce>* to a set of bytes so that they can be taken as inputs of SHA256. You can convert them using String encode*()* method. After the conversion, you can append <Nonce> to <The last block>. (e.g., *str(i).encode()* where *i* is <Nonce> and set as integer.)

3. "Transaction" is the choice of the player in its turn. It contains the player name and its choice of the space on the tic-tac-toe board (e.g., ["Alice",5]). To make your code simple, assume that Alice and Bob take a random empty space in their turn.

For example, Assume that we use the following grid to play a tic-tac-toe game.

1	2	3
4	5	6
7	8	9

The first block (block1.json) is generated by Alice as follows.

```
{
        "TxID": 1,
        "Hash": "00e260d79735cd52f38358289f890019e376117be1866
                 027e172c391383ff5b8",
        "Nonce": 11,
        "Transaction": ["Alice",5]
}
```

Note that in your program, Alice must choose a space, randomly. Then, Alice saves the block in her storage and sends it to Bob.

Once Bob receives the block, Bob validates the received block and save it to its storage. Then, Bob creates a following block and send it to Alice:

```
{
        "TxID": 2,
        "Hash":"0053560513cc0535db6692c972aab41ec30ec9ec3b43a5
                b0b0f1c8eafce20c3f",
        "Nonce": 742,
        "Transaction": ["Bob",1]
}
```

Note that in your program, Bob must randomly choose a space among the spaces not taken previously.

The game will continue **until all spaces on the board are taken**. Alice and Bob keep each block in a separate file (block0.json, block1.json, …, block9.json) in their local storage. Before each player executes their turn, it must check that the block received is valid.

Make the players exchange blocks online. You can use the PubNub package to implement this in Python.

Marks

- Block generation process implementation. (30 marks)
- Block validation process implementation. (10 marks)
- The other basic requirements of the system (e.g., implementation of a tic-tac-toe game and handling JSON format.) (30 marks)
- Implementation of blocks exchange protocol (20 marks)
- Report (10 marks)

Submission

Make a folder named `Assignment3` and include

- Programs for Alice and Bob.
- The history of one of the completed games (e.g., block0.txt, block1.txt, …, and block9.txt)
- Report explaining the requirements to execute your program and the expected outcome of your program.

Compress the `Assignment3` folder using a zip program to create `yourSurname_Assignment3.zip`.

Use Moodle to upload your zip file.