CS251: Cryptocurrencies and Blockchain Technologies **Fall 2023**

Programming Project #1

Due: 11:59pm on Wed., **Oct. 4, 2023** Submit via Gradescope code: **7DVJKY**

In this project you will implement a Python function to generate a Merkle proof. Please download the starter code in `proj1.zip`. In it you will find four files:

- `prover.py`: This script generates the arguments for the verfier and writes them to a file for the verifier to read. Specifically it writes the leaf position, the leaf value and finally the hashes used to prove of the leaf's presence at the given position in the Merkle tree.

  Start python script by running "`python3 prover.py 683`" from the command line. This script first calls the function `gen_leaves_for_merkle_tree()` to generate a thousand strings that will make up the leaves of a Merkle tree. Next it calls the method `gen_merkle_proof()` to generate the hashes for the Merkle proof for leaf number 683 (683 is the number provided on the command line). Finally, it writes the Merkle proof to a text file `merkle_proof.txt`.

  Your job is to implement the function `gen_merkle_proof()`. The missing code in that function can be implemented in less than ten lines of Python.

- `verifier.py`: This script contains a hardcoded Merkle root `ROOT` for the Merkle tree whose leaves were generated by `gen_leaves_for_merkle_tree()`. The script reads in the Merkle proof generated by the prover and verifies that the proof proves the leaf is at the stated position with respect to the hardcoded `ROOT`. **You should not make any changes to this file.** However, you should familiarize yourself with the function `compute_merkle_root_from_proof()`, which is the core of the verifier. This will help you implement the function `gen_merkle_proof()`. Your prover will need to generate a Merkle proof that is accepted by this verifier.

- `merkle_utils.py`: This python script contains helpers to make generating the proof and verifying the proof easier. **You should not make any changes to this file.** However, **you must** familiarise yourself with what each helper function and class does to be able to understand the starter code in `prover.py` and `verifier.py`.

- `proof-for-leaf-95.txt`: An example Merkle proof for leaf #95. Your task is to generate a proof file like this for the leaf specified on the command line.

After you implement the function `gen_merkle_proof()` in `prover.py`, running both scripts one after the other should generate the following output:

```
$ python3 prover.py 683

    I generated 1000 leaves for a Merkle tree of height 10.
    I generated a Merkle proof for leaf #683 in file merkle_proof.txt

$ python3 verifier.py 683

    I verified the Merkle proof: leaf #683 in the committed tree is "data item 683".
```

Try changing one character in `merkle_proof.txt` and check that the verifier now rejects the proof.

Note: there are many implementations of Merkle trees on the web. However, it is more fun, and much more instructive, to implement the prover yourself.

Deliverables. Please submit your file `prover.py` via Gradescope. The autograder will test your prover on a random leaf.

Hints. To aid your understanding of the starter code, try to answer the following questions for your self:

- What does the verifier expect you to include in the `proof`?
- How is `height` defined?
- What is the purpose of the `padding` in `gen_merkle_proof_hashes()`?

You do not need to submit your answers to these questions.

Another helpful exercise is to generate the Merkle proof for leaf number 95 with the command "`python3 prover.py 95`" so you can compare the result to the expected output provided in the file `proof-for-leaf-95.txt`.

Additional reading. We discussed Merkle trees in class, but if you want to read more about them, then this is a good resource.