CS 426: Introduction to Blockchains

Starter Code and Sample Outputs

Part 1: Block and Blockchain Class Implementation

Starter Code

```
#include <iostream>
#include <vector>
#include <string>
#include <sstream>
#include <iomanip>
#include <ctime>
#include <openssl/sha.h>

using namespace std;

// -------------------------
// Block Class
// -------------------------
class Block {
public:
    // TODO: Define the fields for the block (parentHash, nonce, difficulty
        , timestamp, merkleRoot, transactions, hash)

    Block(/* TODO: Add parameters */) {
        // TODO: Initialize the block fields
    }

    // TODO: Implement a function to calculate the hash of the block
    string calculateHash() const {
        // TODO: Combine fields like parentHash, merkleRoot, nonce,
           timestamp into a single string and hash it
        return "";
    }

    // TODO: Implement a static function to calculate the Merkle root from
       transactions
    static string calculateMerkleRoot(const vector<string>& transactions) {
        // TODO: Hash the concatenated transactions
        return "";
    }
};
```

```cpp

// -------------------------
// Blockchain Class
// -------------------------
class Blockchain {
private:
    // TODO: Define a vector or container to store the chain of blocks

public:
    Blockchain() {
        // TODO: Create the Genesis block and add it to the chain
    }

    // TODO: Implement a function to add a new block to the blockchain
    void addBlock(const vector<string>& transactions) {
        // TODO: Use the latest block to generate the next block and add it to the chain
    }

     // TODO: Implement a function to add a new block to the blockchain
    void tip(const vector<string>& transactions) {
        // TODO: Use the latest block to generate the next block and add it to the chain
    }

    // TODO: Implement a function to display block details
    void displayBlock(const Block& block) const {
        // TODO: Print the fields of the block
    }

    // TODO: Implement a function to display the blockchain hashes
    void displayBlockchainHashes() const {
        // TODO: Display the hashes of all blocks in the chain from Genesis to Tip
    }
};

// -------------------------
// Main Function
// -------------------------
int main() {
    // TODO: Create a Blockchain object

    // TODO: Add blocks with dummy transactions to the blockchain

    // TODO: Display the details of the blockchain

    return 0;
}
```

Sample Output

```
Genesis block created with hash: 8
   f5b1caa57738b8d405b621451fefd49e7b8717ddfbacece764dca7f93073731
------ New Block ------
Parent Hash: 8
   f5b1caa57738b8d405b621451fefd49e7b8717ddfbacece764dca7f93073731
Nonce: 1000
Difficulty: 000000000000000000000000000000000000
Timestamp: 1737966107
Merkle Root: 39704
   f929d837dc8bd8e86c70c4fb06cf740e7294f1036d030e92fe545f18275
Hash: 85a1e36ac25cfed3c3f4320cca17467ec20ef98a7c9a6d04a89df7c844d50e1e
Current Blockchain Height: 1
-----------------------------
------ New Block ------
Parent Hash: 85
   a1e36ac25cfed3c3f4320cca17467ec20ef98a7c9a6d04a89df7c844d50e1e
Nonce: 2000
Difficulty: 000000000000000000000000000000000000
Timestamp: 1737966107
Merkle Root: 64833
   afa7026409be938e6e21a643749233e5d418b906fe5b6f304e7a7636eef
Hash: e540859a6f5f27bd7ade9472868916883d7ac3beeab700456a2ad99d8e37fc09
Current Blockchain Height: 2


Blockchain from Genesis to Tip:
Block Hash: 8
   f5b1caa57738b8d405b621451fefd49e7b8717ddfbacece764dca7f93073731
Block Hash: 85
   a1e36ac25cfed3c3f4320cca17467ec20ef98a7c9a6d04a89df7c844d50e1e
Block Hash:
   e540859a6f5f27bd7ade9472868916883d7ac3beeab700456a2ad99d8e37fc09

Blockchain from Tip to Genesis:
Block Hash: 12
   c1c7930b262916deca28b365749f92e149938a1b3d276ba565742db9c027de
Block Hash:
   bd090041a2b235b4f47417eadb7989b9758ee20af03a4198cd352730b06748cb
Block Hash: 8
   fc7be2484378381234c245ecdd58b52e3b6e60e479c4c40a3e5aa73aa5a46bf
```