CS-4049 Blockchain and Cryptocurrency
Project 01

Create a layer 1 Blockchain in golang https://golang.org

It is a group project; however, the size of the group shall not exceed four students. Your submission shall be made on the classroom, and you shall submit a zip file containing all the source code. Deadline is **27th November**. Late submissions or ones sent by email are NOT accepted and WILL give you zero points. You should use the following naming convention while uploading file:

project01_RollNo1_RollNo2_RollNo3_RollNo4.zip

The project consists of two parts named as assignments in the following. The first assignment focuses on the blockchain features related to a single peer. The second assignment is related to network features.

Assignment 1 [100 marks]

1. `Create New Block [10 marks]`
   A method to create a new block. To keep things simple, each transaction can be a simple string of arbitrary length. Among other information, the block header will contain hash pointers to the previous block and the root of the Merkel tree (see below). The nonce value will be initially 0. You can use any hash function of your choice including SHA256.

2. `Create Merkel Tree [40 marks]`
   The blockchain implementation will provide the functionality to create Merket Tree. All the transactions of each block are arranged in a Merkel Tree.

3. `Mine Block [20 marks]`
   A method to find the nonce value for the block. The target shall be adjustable as the number of trailing zeros in the 256-bit output string.

4. `Display Blocks [No separate marks]`
   A method to print all the blocks in a nice format showing block data such as nonce, previous hash, current block hash.

5. `Display Merkel Tree [No separate marks]`
   Function to print all the transactions in a nice format showing the transactions and hashes.

6. `Verify Block and Chain [20 marks]`
   Functionality to verify a block. Moreover, a functionality to verify the blockchain in case any changes are made. The verification will consider the changes to the transactions stored in the Merkel tree.

7. `Change Block [No separate marks]`
   Function to change one or multiple transactions of the given block ref.

8. `Calculate Hash [10 marks]`
   Function for calculating hash of a transaction or a block. If the size of the transaction is very large, then Merkle-Damgard Transform shall be used.

Moreover, your assignment shall include the code to properly demonstrate the functionality of the above features. No marks are awarded if a group is not able to show that the above-mentioned features are working properly.

Assignment 2 [200 marks]

Your blockchain should provide the following network related features:

1. `Create P2P network [40 marks]`
   Nodes in the P2P network can be emulated using goroutines. Each node acts as a server and a client. To provide server functionality, each node listens on a separate port.

2. `Bootstrap node [20 marks]`
   One peer is designated as a bootstrap node. The IP address/port number of the bootstrap node is globally known. The bootstrap node keeps track of the IP addresses/port numbers of all other nodes in the P2P network. In other words, each node in the P2P network registers its IP address/port number with the bootstrap node.

3. `Joining the network [20 marks]`
   A new node is created and joins the network at any time. To join, the new node contacts the bootstrap node to get the IP addresses/port numbers of few existing nodes in the P2P network. The received IP addresses/port numbers are used to establish connections with the neighbors.

4. `Display P2P network [No separate marks]`
   A function to display all the nodes in the P2P network and how they are connected to each other.

5. `Transaction Propagation [40 marks]`
   Each node on receiving a transaction can broadcast it using flooding. Alternatively, students are allowed to implement Gossip protocol. Each node keeps a list of recently received transactions that are not yet included in any valid block. If the same transaction is received twice, it is not added to the list.

6. `Block mining and propagation [40 marks]`
   Once the list contains n transactions, a node creates a block and starts the mining process (already developed in part 1). The mined block is broadcasted to the P2P network using flooding.

7. `Transaction list pruning [20 marks]`
   On receiving a mined block, a node verifies the block (already developed in part 1). If the block is valid, the list of transactions locally maintained by the node is pruned to remove transactions that are included in the mined block.

8. `Block on longest chain [20 marks]`
   Each node strives to accept blocks on the current longest chain. If a node receives a block belonging to the longer chain, it will accept the block and the previous block becomes orphan. If some blocks are missing on the longest chain, the node ask a neighbor to receive the missed blocks.

Moreover, your main shall include the code to properly demonstrate the functionality of the above features. You must be able to demonstrate a P2P network with 10 nodes.

Note: No marks will be awarded if you are not able to demonstrate the functionality of a feature.

Project Grading: In addition to the timely submission of the project on classroom, each group will have to give a 20 mins demonstration. Each member of the group will have to answer individual questions and get a separate score in terms of percentage contributions on the project. The percentage contributions ranges from 0% to 100%, where 0% depicts no contributions at all. The overall marks of each student will be calculated as follows: (percentage contribution) x (group marks in the project). For example, if a group gets 280 marks in the project (out of 300) and a group member receives 50% score as contribution, his/her overall marks will be 140.

Moreover, 25% of the final exam will be comprised of the questions from this project.