Name:________________________________ Email ID:_________________________________
Roll No:______________ College:____________________________ Course:_______________

General Instructions:

1) Please upload the code on your Github account post completion of this assignment
2) Send your Github account link on mail to **campus@posistmail.com**
3) Please return this paper to the invigilator while leaving.

Create a dynamic list of ordered records. A record is owned by an owner. Also, a record has data in the form of a floating point integral value. Through the scheme of symmetric encryption, data is encrypted by the owner and can be decrypted only by the owner.

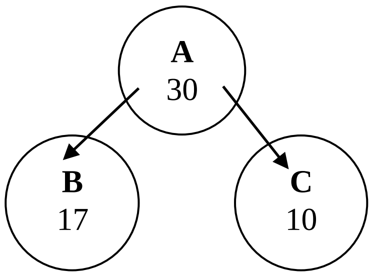
Every node has a parent and some children associated with it. There is a root node called as the ***Genesis Node*** that has no parent associated with it.

Each node of a record has some information associated with it. Structure of the node is as follows:

Key	**Type**
timestamp	Date - universal type that can be formatted to any time zone and any format
data	String - encrypted with the following data values : • Owner id of a node • value (floating point integer fixed upto 2 decimal places) • Owner name • Hash of the set {id, value, name}
nodeNumber	Integer - unique incremental integral value
nodeId	String - 32 bit id that uniquely identifies a node
referenceNodeId	String - address of the parent node
childReferenceNodeId	[String] - addresses of the children
genesisReferenceNodeId	String - address of the genesis node
HashValue	String - hash value of the set {timestamp, data, nodeNumber, nodeId, referenceNodeId, childReferenceNodeId, genesisReferenceNodeId} - hash has preimage resistance, collision resistance and second preimage resistance

The value of the node trickles down from the root node.

For example,



A new node, **D** with its parent as **A,** cannot have a value greater than 30 - (17 + 10) = 3. This is true for all new child nodes that will be created. Sum of a set of value of siblings should never be greater than the value of the root node.

At any point of time, the sum of all values of nodes should never be larger than the value of the genesis node.

Using data structure and programming language of your choice, design an efficient application that realises the following tasks:

1. Create the Genesis Node. Genesis Node has parent node as null.
2. Create set of child nodes of a particular node.
3. Create a child node that originates from a particular node.
4. Encrypt and decrypt the data inside the node.
5. Verify the owner of the node, with the encryption key. Symmetric encryption shows the data. The integrity of the data is checked by computing the hash value of the data and checking with the already computed hash.
6. Edit value of a node. Editing can have many forms, edit value of a subtree root, trickle down the subtree node value to a set of children, or one child or edit value of a leaf node.
7. Transfer ownership of a particular node value. The previous owner validates his/her ownership by decrypting the data, and comparing the hash values. The new owner then encrypts the data with his key.
8. Find the longest chain of the genesis node.
9. Find the longest chain of any node.
10. Merge 2 nodes; **data** is added of the 2 nodes, ownership of the longer chain node are retained after a merge operation.

Instructions

<u>Evaluation Criteria</u>

1. Time taken to deliver the Application. Maximum time is 3 hours.
2. Code Quality - structure and modularity of the code.
3. Creativity - you can add functionality, UI and UX from your side. Mention clearly any assumptions.
4. Limit usage of external libraries. Mention clearly the need of usage of a library.

<u>Submitting Code</u>

1. Create a new account on GitHub, if you don't have.
2. Create a new repository with the source of the code.
3. Create a README.md file to the root of the project that clearly mentions detail of the project and steps to compile it.
4. Send us the link to the created repo.