|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| D:\sep2k3\COLLEG~1\LOGO.JPG | SHRI SANT GAJANAN MAHARAJ COLLEGE OF ENGG. SHEGAON | | | **LABORATORY MANUAL** | |
| **PRACTICAL EXPERIMENT INSTRUCTION SHEET** | | | | |
| LABORATORY MANUAL NO.:SSGMCE/WI/IT/BCF/7IT08/15 | | ISSUE NO.: 00 | | ISSUE DATE: 18/07/2022 | |
| REV.DATE: | | DEPTT:INFORMATION TECHNOLOGY | | | |
| LABORATORY: Block Chain Fundamental | | | SEMESTER: 4N/VII | | PAGE: 1 OF 3 |

**AIM:** Write a program to create chain of at least 3 block containing one Genesis blocks & all others blocks are connected by hash function.

1. **THEORY**

* The system that Bitcoin depends upon - a growing list of records (that is, blocks) that are connected to one another - is called Blockchain. Bitcoin was the first successful application of this system, and soon after its growth in fame, other cryptocurrencies were established on the same beliefs. However, this system is not limited to collecting financial information. Instead, the stored type of data is inconsequential and independent of the Blockchain network.

Fundamentally, the data stored in a blockchain should have consisted of the following characteristics:

* Immutable
* Distributed
* Persistent (no loss of data)
* Unhackable

## Building a blockchain in Python

We have divided the process of building a blockchain into several steps for better understanding. These steps are as follows:

**Step 1:** Creating a Blockchain class

**Step 2:** Writing a Function to build New Blocks

**Step 3:** Writing Functions to create New Transactions and get the Last Block

**Step 4:** Writing a Function to "Hash" the Blocks

**Step 5:** Creating a New Blockchain and Sending some money

1. # importing the required libraries
2. **import** hashlib
3. **import** json
4. from time **import** time
6. # creating the Block\_chain **class**
7. **class** Block\_chain(object):
8. def \_\_init\_\_(self):
9. self.chain = []
10. self.pendingTransactions = []
12. self.newBlock(previousHash = "The Times 03/Jan/2009 Chancellor on brink of second bailout for banks.", the\_proof = 100)
14. # Creating a **new** block listing key/value pairs of
15. # block information in a JSON object.
16. # Reset the list of pending transactions &
17. # append the newest block to the chain.
18. def newBlock(self, the\_proof, previousHash = None):
19. the\_block = {
20. 'index': len(self.chain) + 1,
21. 'timestamp': time(),
22. 'transactions': self.pendingTransactions,
23. 'proof': the\_proof,
24. 'previous\_hash': previousHash or self.hash(self.chain[-1]),
25. }
26. self.pendingTransactions = []
27. self.chain.append(the\_block)
29. **return** the\_block
31. #Searching the blockchain **for** the most recent block.
32. @property
33. def lastBlock(self):
35. **return** self.chain[-1]
37. # Adding a transaction with relevant info to the 'blockpool' - list of pending tx's.
38. def newTransaction(self, the\_sender, the\_recipient, the\_amount):
39. the\_transaction = {
40. 'sender': the\_sender,
41. 'recipient': the\_recipient,
42. 'amount': the\_amount
43. }
44. self.pendingTransactions.append(the\_transaction)
45. **return** self.lastBlock['index'] + 1
47. # receiving one block. Turning it into a string, turning that into
48. # Unicode (**for** hashing). Hashing with SHA256 encryption,
49. # then translating the Unicode into a hexidecimal string.
50. def hash(self, the\_block):
51. stringObject = json.dumps(the\_block, sort\_keys = True)
52. blockString = stringObject.encode()
54. rawHash = hashlib.sha256(blockString)
55. hexHash = rawHash.hexdigest()
57. **return** hexHash
59. block\_chain = Block\_chain()
60. transaction1 = block\_chain.newTransaction("Satoshi", "Alex", '10 BTC')
61. transaction2 = block\_chain.newTransaction("Alex", "Satoshi", '2 BTC')
62. transaction3 = block\_chain.newTransaction("Satoshi", "James", '10 BTC')
63. block\_chain.newBlock(10123)
65. transaction4 = block\_chain.newTransaction("Alex", "Lucy", '2 BTC')
66. transaction5 = block\_chain.newTransaction("Lucy", "Justin", '1 BTC')
67. transaction6 = block\_chain.newTransaction("Justin", "Alex", '1 BTC')
68. block\_chain.newBlock(10384)
70. print("Genesis block: ", block\_chain.chain)

**03 CONCLUSION:**. In this way, we create chain of at least 3 block containing one Genesis blocks & all others blocks are connected by hash function.