Practical 6 Add blocks to the miner and dump it

In [1]:

```
import datetime
import collections
import binascii
import Crypto  #If not installed run pip install crypto and pip install pycryptodome
import Crypto.Random
from Crypto.Hash import SHA
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5
```

In [2]:

```
def display_transaction(transaction):
    #for transaction in transactions:
    dict = transaction.to_dict()
    print ("sender: " + dict['sender'])
    print ('-----')
    print ("recipient: " + dict['recipient'])
    print ('-----')
    print ("value: " + str(dict['value']))
    print ('-----')
    print ("time: " + str(dict['time']))
    print ('-----')
```

In [3]:

```
class Block:
    def __init__(self):
        self.verified_transactions=[]
        self.previous_block_hash=""
        self.Nounce=""
```

In [4]:

```
last_block_hash=""
```

In [5]:

```
class Client:
    def __init__(self):
        random = Crypto.Random.new().read
        self._private_key = RSA.generate(1024, random)
        self._public_key = self._private_key.publickey()
        self._signer = PKCS1_v1_5.new(self._private_key)

    @property
    def identity(self):
        return binascii.hexlify(self._public_key.exportKey(format='DER')).decode("ascii")
```

In [6]:

```
Alice=Client()
```

In [7]:

```
class Transaction:
    def __init__(self,sender,recipient, value):
        self.sender=sender
        self.recipient=recipient
        self.value=value
        self.time=datetime.datetime.now()
    def to_dict(self):
        if self.sender == "Genesis":
            identity = "Genesis"
        else:
            identity = self.sender.identity
        #Dictionary objects
        return collections.OrderedDict({
            'sender':identity,
            'recipient':self.recipient,
            'value':self.value,
            'time':self.time
    })
    def sign_transaction(self):
        private_key = self.sender._private_key
        signer = PKCS1_v1_5.new(private_key)
        h = SHA.new(str(self.to_dict()).encode('utf8'))
        return binascii.hexlify(signer.sign(h)).decode('ascii')
```

In [8]:

```
t0=Transaction("Genesis",Alice.identity,400.0)
```

In [9]:

```
block0=Block()
```

In [10]:

```
block0.previous_block_hash=None
Nounce = None
```

In [11]:

```
block0.verified_transactions.append(t0)
```

In [12]:

```
digest=hash(block0)
last_block_hash=digest
```

In [13]:

```
digest
```

Out[13]:

```
157158626332
```

In [14]:

```
last_block_hash
```

Out[14]:

```
157158626332
```

In [15]:

```
MyBlockchain1 = []
```

In [16]:

```
def dump_blockchain (self):
    print ("Number of blocks in the chain: " + str(len (self)))
    for x in range (len(MyBlockchain1)):
        block_temp = MyBlockchain1[x]
        print ("block # " + str(x))
        for transaction in block_temp.verified_transactions:
            display_transaction (transaction)
            print ('--------------')
        print ('====================================')
```

In [17]:

```
MyBlockchain1.append(block0)
```

In [18]:

```
dump_blockchain(MyBlockchain1)
```

```
Number of blocks in the chain: 1
block # 0
sender: Genesis
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100a663816
d24d6ca2b97b2a8903d8671f6e4cd6ac787a097db4ef41ed2139189db1951d7065c1dc05da7a
4b01ed469185b3e3e847a42f38c49adb3a4704f8dcbec63efb7ec3b4afc6e5cd82d26919fb05
1dc4f1929e75eebdfaa5759d8ddfe16b197eea9566c4b44ddc7f095ad29580f45fd131595ffc
5fe61211e5deaf9704cc90203010001
-----
value: 400.0
-----
time: 2022-06-16 10:35:35.137102
-----
--------------
====================================
```

In [ ]:

```
```