Name : Vaibhav jha
Roll no : 118
Course : MSC-CS-5
Subject : Blockchain Technology – Practical

Assignment -1 : Blockchain on retail sector

```
import time

import hashlib

from typing import Any,Dict


class Block:
   def __init__(self,index:int,data:Dict[str,Any],previous_hash:str)->None:
      self.index=index
      self.timestamp=time.time()
      self.data=data
      self.previous_hash=previous_hash
      self.nonce=0
      self.hash=self.compute_hash()

   def compute_hash(self)->str:
      block_string = f"{self.index}{self.timestamp}{self.data}{self.previous_hash}{self.nonce}"
      print(block_string)
      return hashlib.sha256(block_string.encode()).hexdigest()

   def mine_block(self,difficulty:int)->None:
      target="0" * difficulty
      while not self.hash.startswith(target):
         self.nonce=self.nonce+1
         self.hash=self.compute_hash()

   def __repr__(self)->str:
        return (f"Block(index={self.index}, timestamp={self.timestamp}, "
                f"data={self.data}, previous_hash={self.previous_hash}, "
                f"hash={self.hash}, nonce={self.nonce})")

block = Block(index=1,data={"sender":"Alice","reciever":"Bob","Amount":100},previous_hash="0")

print(repr(block))

print()

print(block)

print("Index:", block.index)
print("Timestamp:", block.timestamp)
print("Data:", block.data)
print("Previous Hash:", block.previous_hash)
print("Current Hash:", block.hash)

class Blockchain:
    def __init__(self):
        self.chain = []
        self.create_genesis_block()

    def create_genesis_block(self):
        genesis_block = Block(0, {"message": "Genesis Block"}, "0")
        self.chain.append(genesis_block)

    def add_block(self, transactions):
        previous_block = self.chain[-1]
        new_block = Block(len(self.chain), transactions, previous_block.hash)
        self.chain.append(new_block)
        return new_block
    def get_all_blocks(self):
        return self.chain

    def find_block_by_index(self, index):
        return next((block for block in self.chain if block.index == index), None)

    def find_block_by_hash(self, hash_value):
        return next((block for block in self.chain if block.hash == hash_value), None)

blockchain = Blockchain()

while True:
    print("\n--- Blockchain Explorer ---")
    print("1. Add Block")
    print("2. View All Blocks")
    print("3. Find Block by Index")
    print("4. Find Block by Hash")
    print("5. Exit")

    choice = input("Enter choice: ")

    if choice == "1":
        sender = input("Sender: ")
        receiver = input("Receiver: ")
        amount = input("Amount: ")
        block = blockchain.add_block({"sender": sender, "receiver": receiver, "amount": amount})
        print("Block added:", block)

    elif choice == "2":
        for block in blockchain.get_all_blocks():
            print(block)

    elif choice == "3":
        index = int(input("Enter index: "))
        block = blockchain.find_block_by_index(index)
        print(block if block else "Block not found")

    elif choice == "4":
        hash_value = input("Enter hash: ")
        block = blockchain.find_block_by_hash(hash_value)
        print(block if block else "Block not found")

    elif choice == "5":
        break
```

Output:

```
01763392983.6820781{'message': 'Genesis Block'}00

--- Blockchain Explorer ---
1. Add Block
2. View All Blocks
3. Find Block by Index
4. Find Block by Hash
5. Exit
Enter choice:  1
Sender:  abc
Receiver:  def
Amount:  10000
11763393114.2380388{'sender': 'abc', 'receiver': 'def', 'amount':
'10000'}e498bb8ed16411abf752db8368befb03642c25c7435c00c2d46c3614d7782de00
```

```
Block added: Block(index=1, timestamp=1763393114.2380388, data={'sender': 'abc', 'receiver': 'def',
'amount': '10000'},
previous_hash=e498bb8ed16411abf752db8368befb03642c25c7435c00c2d46c3614d7782de0,
hash=01d245406f82ddbf06a23f756abfc73bcfa403ecd80b5e87cbb2fd629718c281, nonce=0)


--- Blockchain Explorer ---

1. Add Block

2. View All Blocks

3. Find Block by Index

4. Find Block by Hash

5. Exit

Enter choice:  2

Block(index=0, timestamp=1763392983.6820781, data={'message': 'Genesis Block'},
previous_hash=0,
hash=e498bb8ed16411abf752db8368befb03642c25c7435c00c2d46c3614d7782de0, nonce=0)

Block(index=1, timestamp=1763393114.2380388, data={'sender': 'abc', 'receiver': 'def', 'amount':
'10000'},
previous_hash=e498bb8ed16411abf752db8368befb03642c25c7435c00c2d46c3614d7782de0,
hash=01d245406f82ddbf06a23f756abfc73bcfa403ecd80b5e87cbb2fd629718c281, nonce=0)


--- Blockchain Explorer ---

1. Add Block

2. View All Blocks

3. Find Block by Index

4. Find Block by Hash

5. Exit

Enter choice:  3

Enter index:  1

Block(index=1, timestamp=1763393114.2380388, data={'sender': 'abc', 'receiver': 'def', 'amount':
'10000'},
previous_hash=e498bb8ed16411abf752db8368befb03642c25c7435c00c2d46c3614d7782de0,
hash=01d245406f82ddbf06a23f756abfc73bcfa403ecd80b5e87cbb2fd629718c281, nonce=0)


--- Blockchain Explorer ---

1. Add Block
```

```
2. View All Blocks
3. Find Block by Index
4. Find Block by Hash
5. Exit
Enter choice:  5
```

Assignment -2 : Bank Sector Blockchain

```
import hashlib
import json
import time

class Block:
    def __init__(self, index, transaction, previous_hash):
        self.index = index
        self.timestamp = time.time()
        self.transaction = transaction
        self.previous_hash = previous_hash
        self.hash = self.compute_hash()

    def compute_hash(self):
        block_string = json.dumps({
            "index": self.index,
            "timestamp": self.timestamp,
            "transaction": self.transaction,
            "previous_hash": self.previous_hash
        }, sort_keys=True).encode()

        return hashlib.sha256(block_string).hexdigest()

    def __str__(self):
        return (f"Block(Index: {self.index}, Hash: {self.hash}, "
                f"Prev Hash: {self.previous_hash}, Transaction: {self.transaction}, "
                f"Timestamp: {self.timestamp})")

class BankingBlockchain:
    def __init__(self):
        self.chain = []
        self.create_genesis_block()

    def create_genesis_block(self):
        genesis_block = Block(0, {"message": "Genesis Block"}, "0")
        self.chain.append(genesis_block)

    def add_transaction(self, transaction_type, account_id, amount):
        previous_block = self.chain[-1]
        transaction = {
            "type": transaction_type,
            "account_id": account_id,
            "amount": amount
        }
        new_block = Block(len(self.chain), transaction, previous_block.hash)
        self.chain.append(new_block)
        return new_block

    def get_all_blocks(self):
        return self.chain
    def find_block_by_index(self, index):
        return next((block for block in self.chain if block.index == index), None)

    def find_block_by_hash(self, hash_value):
        return next((block for block in self.chain if block.hash == hash_value), None)

bank_chain = BankingBlockchain()

while True:
    print("\n--- Banking Blockchain ---")
    print("1. Create Account")
    print("2. Deposit Money")
    print("3. Withdraw Money")
    print("4. Process Loan")
    print("5. View All Transactions")
    print("6. Find Transaction by Index")
    print("7. Find Transaction by Hash")
    print("8. Exit")

    choice = input("Enter choice: ")

    if choice == "1":
        account_id = input("Enter New Account ID: ")
        bank_chain.add_transaction("account_creation", account_id, 0)
        print("Account Created Successfully.")

    elif choice == "2":
        account_id = input("Enter Account ID: ")
        amount = float(input("Enter Deposit Amount: "))
        bank_chain.add_transaction("deposit", account_id, amount)
        print(f"Deposited {amount} to {account_id}")

    elif choice == "3":
        account_id = input("Enter Account ID: ")
        amount = float(input("Enter Withdrawal Amount: "))
        bank_chain.add_transaction("withdrawal", account_id, amount)
        print(f"Withdrawn {amount} from {account_id}")

    elif choice == "4":
        account_id = input("Enter Account ID: ")
        amount = float(input("Enter Loan Amount: "))
        bank_chain.add_transaction("loan", account_id, amount)
        print(f"Loan of {amount} processed for {account_id}")

    elif choice == "5":
        for block in bank_chain.get_all_blocks():
            print(block)

    elif choice == "6":
        index = int(input("Enter index: "))
        block = bank_chain.find_block_by_index(index)
        print(block if block else "Transaction not found")

    elif choice == "7":
        hash_value = input("Enter hash: ")
        block = bank_chain.find_block_by_hash(hash_value)
        print(block if block else "Transaction not found")

    elif choice == "8":
        break
```

```
--- Banking Blockchain ---
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Process Loan
5. View All Transactions
6. Find Transaction by Index
7. Find Transaction by Hash
8. Exit
Enter choice:  1
Enter New Account ID:  123456
Account Created Successfully.

--- Banking Blockchain ---
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Process Loan
5. View All Transactions
6. Find Transaction by Index
7. Find Transaction by Hash
8. Exit
Enter choice:  2
Enter Account ID:  123456
Enter Deposit Amount:  12000
Deposited 12000.0 to 123456

--- Banking Blockchain ---
1. Create Account
2. Deposit Money
3. Withdraw Money
```

```
4. Process Loan

5. View All Transactions

6. Find Transaction by Index

7. Find Transaction by Hash

8. Exit

Enter choice:  5

Block(Index: 0, Hash: 850ef467bf2f08505a4d12c248557e6d00099df480076025ab8e7e265dd75865,
Prev Hash: 0, Transaction: {'message': 'Genesis Block'}, Timestamp: 1758541858.8979192)

Block(Index: 1, Hash: 1618557523797edda58bf8ab328da7878a4e2f5a6aaf60a19112f2e205265794,
Prev Hash: 850ef467bf2f08505a4d12c248557e6d00099df480076025ab8e7e265dd75865,
Transaction: {'type': 'account_creation', 'account_id': '123456', 'amount': 0}, Timestamp:
1758541868.7275386)

Block(Index: 2, Hash: bf382716ac68a44220dbb3d863ef77b76d67a79f13fb876a7b2dc3cbe42a5cef,
Prev Hash: 1618557523797edda58bf8ab328da7878a4e2f5a6aaf60a19112f2e205265794,
Transaction: {'type': 'deposit', 'account_id': '123456', 'amount': 12000.0}, Timestamp:
1758541876.211505)


--- Banking Blockchain ---

1. Create Account

2. Deposit Money

3. Withdraw Money

4. Process Loan

5. View All Transactions

6. Find Transaction by Index

7. Find Transaction by Hash

8. Exit
```

Assignment – 3 : Blockchain in Medical Sector

```
import hashlib
import json

import time


class MedicalBlock:

    def __init__(self, index, record, previous_hash):

        self.index = index

        self.timestamp = time.time()

        self.record = record  # Medical record transaction details

        self.previous_hash = previous_hash

        self.hash = self.compute_hash()


    def compute_hash(self):

        block_string = json.dumps({

            "index": self.index,

            "timestamp": self.timestamp,

            "record": self.record,

            "previous_hash": self.previous_hash

        }, sort_keys=True).encode()


        return hashlib.sha256(block_string).hexdigest()


    def __str__(self):

        return (f"Block(Index: {self.index}, Hash: {self.hash}, Prev Hash: {self.previous_hash}, "

                f"Record: {self.record}, Timestamp: {self.timestamp})")


class MedicalBlockchain:

    def __init__(self):

        self.chain = []

        self.create_genesis_block()


    def create_genesis_block(self):
        genesis_block = MedicalBlock(0, {"message": "Genesis Medical Record Block"}, "0")
        self.chain.append(genesis_block)

    def add_record(self, record_type, patient_id, details):
        previous_block = self.chain[-1]
        record = {
            "type": record_type,
            "patient_id": patient_id,
            "details": details
        }
        new_block = MedicalBlock(len(self.chain), record, previous_block.hash)
        self.chain.append(new_block)
        return new_block

    def get_all_blocks(self):
        return self.chain

    def find_block_by_index(self, index):
        return next((block for block in self.chain if block.index == index), None)

    def find_block_by_hash(self, hash_value):
        return next((block for block in self.chain if block.hash == hash_value), None)

# Example usage in a menu-driven format
medical_chain = MedicalBlockchain()

while True:
    print("\n--- Medical Blockchain ---")
    print("1. Register Patient")
    print("2. Add Doctor Visit")
    print("3. Add Prescription")
    print("4. Add Test Result")
    print("5. View All Medical Records")
    print("6. Find Record by Index")
    print("7. Find Record by Hash")
    print("8. Exit")

    choice = input("Enter choice: ")

    if choice == "1":
        patient_id = input("Enter New Patient ID: ")
        name = input("Enter Patient Name: ")
        dob = input("Enter Date of Birth (YYYY-MM-DD): ")
        record_details = {"name": name, "dob": dob}
        medical_chain.add_record("patient_registration", patient_id, record_details)
        print("Patient Registered Successfully.")

    elif choice == "2":
        patient_id = input("Enter Patient ID: ")
        doctor = input("Enter Doctor Name: ")
        visit_date = input("Enter Visit Date (YYYY-MM-DD): ")
        notes = input("Enter Visit Notes: ")
        record_details = {"doctor": doctor, "visit_date": visit_date, "notes": notes}
        medical_chain.add_record("doctor_visit", patient_id, record_details)
        print("Doctor Visit Added.")

    elif choice == "3":
        patient_id = input("Enter Patient ID: ")
        medication = input("Enter Medication Details: ")
        dosage = input("Enter Dosage Information: ")
        record_details = {"medication": medication, "dosage": dosage}
        medical_chain.add_record("prescription", patient_id, record_details)
```

```
        print("Prescription Added.")

    elif choice == "4":
        patient_id = input("Enter Patient ID: ")
        test_name = input("Enter Test Name: ")
        test_date = input("Enter Test Date (YYYY-MM-DD): ")
        result = input("Enter Test Result: ")
        record_details = {"test_name": test_name, "test_date": test_date, "result": result}
        medical_chain.add_record("test_result", patient_id, record_details)
        print("Test Result Added.")

    elif choice == "5":
        for block in medical_chain.get_all_blocks():
            print(block)

    elif choice == "6":
        index = int(input("Enter index: "))
        block = medical_chain.find_block_by_index(index)
        print(block if block else "Record not found")

    elif choice == "7":
        hash_value = input("Enter hash: ")
        block = medical_chain.find_block_by_hash(hash_value)
        print(block if block else "Record not found")

    elif choice == "8":
        break
```

```
--- Medical Blockchain ---
```

```
1. Register Patient

2. Add Doctor Visit

3. Add Prescription

4. Add Test Result

5. View All Medical Records

6. Find Record by Index

7. Find Record by Hash

8. Exit

Enter choice:  1

Enter New Patient ID:  101

Enter Patient Name:  abc

Enter Date of Birth (YYYY-MM-DD):  2005-07-22

Patient Registered Successfully.


--- Medical Blockchain ---

1. Register Patient

2. Add Doctor Visit

3. Add Prescription

4. Add Test Result

5. View All Medical Records

6. Find Record by Index

7. Find Record by Hash

8. Exit

Enter choice:  5

Block(Index: 0, Hash: 651c13192e7090567cc2008c3afdf941925c2d7c24242cfc6e2ab9e4c79aeb9a,
Prev Hash: 0, Record: {'message': 'Genesis Medical Record Block'}, Timestamp:
1763393585.3745303)

Block(Index: 1, Hash: d79f608af416255039526890654d0091e8ccb319b4d9a73c19a5387860ea624a,
Prev Hash: 651c13192e7090567cc2008c3afdf941925c2d7c24242cfc6e2ab9e4c79aeb9a, Record:
{'type': 'patient_registration', 'patient_id': '101', 'details': {'name': 'abc', 'dob': '2005-07-22'}},
Timestamp: 1763393599.0511308)


--- Medical Blockchain ---
```

```
1. Register Patient

2. Add Doctor Visit

3. Add Prescription

4. Add Test Result

5. View All Medical Records

6. Find Record by Index

7. Find Record by Hash

8. Exit

Enter choice:  2

Enter Patient ID:  101

Enter Doctor Name:  def

Enter Visit Date (YYYY-MM-DD):  2025-02-08

Enter Visit Notes:  cold and cough

Doctor Visit Added.


--- Medical Blockchain ---

1. Register Patient

2. Add Doctor Visit

3. Add Prescription

4. Add Test Result

5. View All Medical Records

6. Find Record by Index

7. Find Record by Hash

8. Exit

Enter choice:  5

Block(Index: 0, Hash: 651c13192e7090567cc2008c3afdf941925c2d7c24242cfc6e2ab9e4c79aeb9a,
Prev Hash: 0, Record: {'message': 'Genesis Medical Record Block'}, Timestamp:
1763393585.3745303)

Block(Index: 1, Hash: d79f608af416255039526890654d0091e8ccb319b4d9a73c19a5387860ea624a,
Prev Hash: 651c13192e7090567cc2008c3afdf941925c2d7c24242cfc6e2ab9e4c79aeb9a, Record:
{'type': 'patient_registration', 'patient_id': '101', 'details': {'name': 'abc', 'dob': '2005-07-22'}},
Timestamp: 1763393599.0511308)
```

```
Block(Index: 2, Hash: 469d28629ab14e1b96f660775307a1000ba9603a6f5bfa1979ef2e4a565427f9,
Prev Hash: d79f608af416255039526890654d0091e8ccb319b4d9a73c19a5387860ea624a, Record:
{'type': 'doctor_visit', 'patient_id': '101', 'details': {'doctor': 'def', 'visit_date': '2025-02-08', 'notes':
'cold and cough'}}, Timestamp: 1763393637.0697193)


--- Medical Blockchain ---

1. Register Patient

2. Add Doctor Visit

3. Add Prescription

4. Add Test Result

5. View All Medical Records

6. Find Record by Index

7. Find Record by Hash

8. Exit

Enter choice:  8
```

Assignment – 4 : Energy and utility sector

```
import hashlib

import json

import time


class EnergyBlock:

    def __init__(self, index, transaction, previous_hash):

        self.index = index

        self.timestamp = time.time()

        self.transaction = transaction  # Energy transaction details
        self.previous_hash = previous_hash
        self.hash = self.compute_hash()

    def compute_hash(self):
        block_string = json.dumps({
            "index": self.index,
            "timestamp": self.timestamp,
            "transaction": self.transaction,
            "previous_hash": self.previous_hash
        }, sort_keys=True).encode()

        return hashlib.sha256(block_string).hexdigest()

    def __str__(self):
        return (f"Block(Index: {self.index}, Hash: {self.hash}, Prev Hash: {self.previous_hash}, "
                f"Transaction: {self.transaction}, Timestamp: {self.timestamp})")

class EnergyBlockchain:
    def __init__(self):
        self.chain = []
        self.create_genesis_block()

    def create_genesis_block(self):
        genesis_block = EnergyBlock(0, {"message": "Genesis Energy Block"}, "0")
        self.chain.append(genesis_block)

    def add_transaction(self, transaction_type, participant_id, details):
        previous_block = self.chain[-1]
        transaction = {
            "type": transaction_type,
            "participant_id": participant_id,
            "details": details
        }
        new_block = EnergyBlock(len(self.chain), transaction, previous_block.hash)
        self.chain.append(new_block)
        return new_block

    def get_all_blocks(self):
        return self.chain

    def find_block_by_index(self, index):
        return next((block for block in self.chain if block.index == index), None)

    def find_block_by_hash(self, hash_value):
        return next((block for block in self.chain if block.hash == hash_value), None)


energy_chain = EnergyBlockchain()

while True:
    print("\n--- Energy & Utility Blockchain ---")
    print("1. Register Participant")
    print("2. Record Energy Production")
    print("3. Record Energy Consumption")
    print("4. Issue Renewable Energy Certificate")
    print("5. Record Utility Bill Payment")
    print("6. View All Transactions")
    print("7. Find Transaction by Index")
    print("8. Find Transaction by Hash")
    print("9. Exit")

    choice = input("Enter choice: ")
    if choice == "1":
        participant_id = input("Enter Participant ID (e.g. household, company): ")
        name = input("Enter Participant Name: ")
        record_details = {"name": name}
        energy_chain.add_transaction("participant_registration", participant_id, record_details)
        print("Participant Registered Successfully.")

    elif choice == "2":
        participant_id = input("Enter Producer Participant ID: ")
        energy_amount = float(input("Enter Energy Produced (kWh): "))
        production_time = input("Enter Production Time (YYYY-MM-DD HH:MM): ")
        record_details = {"energy_produced_kWh": energy_amount, "production_time":
production_time}
        energy_chain.add_transaction("energy_production", participant_id, record_details)
        print("Energy Production Recorded.")

    elif choice == "3":
        participant_id = input("Enter Consumer Participant ID: ")
        energy_amount = float(input("Enter Energy Consumed (kWh): "))
        consumption_time = input("Enter Consumption Time (YYYY-MM-DD HH:MM): ")
        record_details = {"energy_consumed_kWh": energy_amount, "consumption_time":
consumption_time}
        energy_chain.add_transaction("energy_consumption", participant_id, record_details)
        print("Energy Consumption Recorded.")

    elif choice == "4":
        participant_id = input("Enter Participant ID to issue certificate: ")
        certificate_id = input("Enter REC Certificate ID: ")
        issue_date = input("Enter Issue Date (YYYY-MM-DD): ")
        record_details = {"certificate_id": certificate_id, "issue_date": issue_date}
        energy_chain.add_transaction("renewable_energy_certificate", participant_id, record_details)
        print("Renewable Energy Certificate Issued.")

    elif choice == "5":
        participant_id = input("Enter Participant ID: ")
        bill_amount = float(input("Enter Bill Amount: "))
        payment_date = input("Enter Payment Date (YYYY-MM-DD): ")
        record_details = {"bill_amount": bill_amount, "payment_date": payment_date}
        energy_chain.add_transaction("utility_bill_payment", participant_id, record_details)
        print("Utility Bill Payment Recorded.")

    elif choice == "6":
        for block in energy_chain.get_all_blocks():
            print(block)

    elif choice == "7":
        index = int(input("Enter index: "))
        block = energy_chain.find_block_by_index(index)
        print(block if block else "Transaction not found")

    elif choice == "8":
        hash_value = input("Enter hash: ")
        block = energy_chain.find_block_by_hash(hash_value)
        print(block if block else "Transaction not found")

    elif choice == "9":
        break
```

Output:

```
--- Energy & Utility Blockchain ---

1. Register Participant

2. Record Energy Production

3. Record Energy Consumption

4. Issue Renewable Energy Certificate

5. Record Utility Bill Payment

6. View All Transactions

7. Find Transaction by Index

8. Find Transaction by Hash

9. Exit

Enter choice:  1

Enter Participant ID (e.g. household, company):  101

Enter Participant Name:  abc

Participant Registered Successfully.


--- Energy & Utility Blockchain ---

1. Register Participant

2. Record Energy Production

3. Record Energy Consumption

4. Issue Renewable Energy Certificate

5. Record Utility Bill Payment

6. View All Transactions

7. Find Transaction by Index

8. Find Transaction by Hash

9. Exit

Enter choice:  2

Enter Producer Participant ID:  101

Enter Energy Produced (kWh):  10000

Enter Production Time (YYYY-MM-DD HH:MM):  2025-03-22 22:30
```

```
Energy Production Recorded.


--- Energy & Utility Blockchain ---
1. Register Participant
2. Record Energy Production
3. Record Energy Consumption
4. Issue Renewable Energy Certificate
5. Record Utility Bill Payment
6. View All Transactions
7. Find Transaction by Index
8. Find Transaction by Hash
9. Exit
Enter choice:


--- Energy & Utility Blockchain ---
1. Register Participant
2. Record Energy Production
3. Record Energy Consumption
4. Issue Renewable Energy Certificate
5. Record Utility Bill Payment
6. View All Transactions
7. Find Transaction by Index
8. Find Transaction by Hash
9. Exit
Enter choice:  6
Block(Index: 0, Hash: 56027d7fcceda5388904341120c95c383968e56e674a7e5d6b6086e6d1f69fa8,
Prev Hash: 0, Transaction: {'message': 'Genesis Energy Block'}, Timestamp: 1763393792.0969942)
Block(Index: 1, Hash: 05e760fb3da1352438f3f52800319745df357c93c3d0b87ed82e04362c405266,
Prev Hash: 56027d7fcceda5388904341120c95c383968e56e674a7e5d6b6086e6d1f69fa8,
Transaction: {'type': 'participant_registration', 'participant_id': '101', 'details': {'name': 'abc'}},
Timestamp: 1763393798.450503)
Block(Index: 2, Hash: 15e7bc8c2db6233689d652bedb6fcb7a8b0970255f18bcdcd213404f719ac5bb,
Prev Hash: 05e760fb3da1352438f3f52800319745df357c93c3d0b87ed82e04362c405266,
Transaction: {'type': 'energy_production', 'participant_id': '101', 'details': {'energy_produced_kWh':
10000.0, 'production_time': '2025-03-22 22:30'}}, Timestamp: 1763393822.4553854)


--- Energy & Utility Blockchain ---
1. Register Participant
2. Record Energy Production
3. Record Energy Consumption
4. Issue Renewable Energy Certificate
5. Record Utility Bill Payment
6. View All Transactions
7. Find Transaction by Index
8. Find Transaction by Hash
9. Exit
Enter choice:  9
```

---END OF ASSIGNMENTS-----------------------------------