In [ ]:
```
import datetime
import hashlib
import json

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

    def create_genesis_block(self):
        block = {
            "index": 1,
            "timestamp": str(datetime.datetime.now()),
            "proof": 1,
            "transactions": [],
            "previous_hash": "0"
        }
        self.chain.append(block)

    def create_block(self, proof, transactions, date, productno, productname, price,
        previous_block = self.chain[-1]
        block = {
            "index": len(self.chain) + 1,
            "timestamp": str(datetime.datetime.now()),
            "proof": proof,
            "transactions": transactions,
            "date": date,
            "productno": productno,
            "productname": productname,
            "price": price,
            "quantity": quantity,
            "customerno": customerno,
            "country": country,
            "previous_hash": self.hash(previous_block)
        }
        self.chain.append(block)

    @staticmethod
    def hash(block):
        block_string = json.dumps(block, sort_keys=True).encode()
        return hashlib.sha256(block_string).hexdigest()

# Example usage:
blockchain = SIDblockchain()

# Get input from the user for each field
transaction_no = input("Enter Transaction Number: ")
date = input("Enter Date (DD-MM-YYYY): ")
product_no = input("Enter Product Number: ")
product_name = input("Enter Product Name: ")
price = float(input("Enter Price: "))
quantity = int(input("Enter Quantity: "))
customer_no = input("Enter Customer Number: ")
country = input("Enter Country: ")

# Create a new block with the user input
blockchain.create_block(1, [], date, product_no, product_name, price, quantity, cust

# Print the updated blockchain
print(json.dumps(blockchain.chain, indent=4))
```