Blockchain Technology Lab

Lab – 2

Aim : Implement SHA-1 and apply it on Doubly Linked List data

Code:

```
import json
import struct

# Node class for the doubly linked list
class Node:
    def __init__(self, admission_number, marks, branch):
        self.admission_number = admission_number
        self.marks = marks
        self.branch = branch
        self.prev = None
        self.next = None

# Doubly Linked List class
class DoublyLinkedList:
    def __init__(self):
        self.head = None

    def append(self, admission_number, marks, branch):
        new_node = Node(admission_number, marks, branch)
        if not self.head:
            self.head = new_node
            return
        temp = self.head
        while temp.next:
            temp = temp.next
        temp.next = new_node
        new_node.prev = temp

    def to_dict(self):
        """Convert the linked list to a list of dictionaries for JSON
serialization."""
        result = []
        temp = self.head
        while temp:
            result.append({
                "admission_number": temp.admission_number,
                "marks": temp.marks,
                "branch": temp.branch
            })
            temp = temp.next
        return result
```

```
    def sha1(self):
        """Compute SHA-1 hash of the concatenated data in the linked list."""
        concatenated_data = b""  # Use bytes for concatenation
        temp = self.head
        while temp:
            concatenated_data +=
f"{temp.admission_number}{temp.marks}{temp.branch}".encode('utf-8')
            temp = temp.next

        return sha1_hash(concatenated_data)

# SHA-1 implementation
def sha1_hash(data):
    # Initial hash values
    h = [
        0x67452301,
        0xEFCDAB89,
        0x98BADCFE,
        0x10325476,
        0xC3D2E1F0
    ]

    # Preprocessing
    original_byte_len = len(data)
    original_bit_len = original_byte_len * 8

    # Append the '1' bit
    data += b'\x80'
    while (len(data) * 8) % 512 != 448:
        data += b'\x00'

    # Append the original length as a 64-bit big-endian integer
    data += struct.pack('>Q', original_bit_len)

    # Process each 512-bit chunk
    for i in range(0, len(data), 64):
        chunk = data[i:i + 64]
        w = list(struct.unpack('>16L', chunk)) + [0] * 64  # Adjusted to 64

        for j in range(16, 80):
            w[j] = left_rotate(w[j - 3] ^ w[j - 8] ^ w[j - 14] ^ w[j - 16], 1)

        a, b, c, d, e = h

        for j in range(80):
            if 0 <= j <= 19:
                f = (b & c) | (~b & d)
                k = 0x5A827999
```

```
            elif 20 <= j <= 39:
                f = b ^ c ^ d
                k = 0x6ED9EBA1
            elif 40 <= j <= 59:
                f = (b & c) | (b & d) | (c & d)
                k = 0x8F1BBCDC
            else:
                f = b ^ c ^ d
                k = 0xCA62C1D6

            temp = (left_rotate(a, 5) + f + e + k + w[j]) & 0xFFFFFFFF
            e = d
            d = c
            c = left_rotate(b, 30)
            b = a
            a = temp

        # Add the compressed chunk to the current hash value
        h = [(x + y) & 0xFFFFFFFF for x, y in zip(h, [a, b, c, d, e])]

    # Produce the final hash value (big-endian) as a 40-digit hexadecimal number
    return ''.join(f'{x:08x}' for x in h)

# Rotate left operation
def left_rotate(n, b):
    return ((n << b) | (n >> (32 - b))) & 0xFFFFFFFF

# Example usage
if __name__ == "__main__":
    dll = DoublyLinkedList()

    # Sample data
    dll.append(101, 85, "CS")
    dll.append(102, 90, "IT")
    dll.append(103, 78, "ME")

    # Compute SHA-1 hash
    sha1_hash_value = dll.sha1()
    print(f"SHA-1 Hash: {sha1_hash_value}")

    # Convert to JSON and save to file
    linked_list_data = dll.to_dict()
    with open("linked_list_data.json", "w") as json_file:
        json.dump(linked_list_data, json_file, indent=4)

    print("Linked list data saved to 'linked_list_data.json'.")
```

Output:

```
  python -u "d:\College\SEM 7\Blockchain LAB\exp 1\sha_with_ll.py"
SHA-1 Hash: 94b516aa8d8d6cb1654580266fd14a1611370cdc
Linked list data saved to 'linked_list_data.json'.
```