import hashlib

def sha256(message):

# Initialize hash values

h0 = 0x6a09e667

h1 = 0xbb67ae85

h2 = 0x3c6ef372

h3 = 0xa54ff53a

h4 = 0x510e527f

h5 = 0x9b05688c

h6 = 0x1f83d9ab

h7 = 0x5be0cd19

# Initialize array of round constants

k = [

0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5,

0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5,

0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3,

0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174,

0xe49b69c1, 0xefbe4786, 0x0fc19dc6, 0x240ca1cc,

0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da,

0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7,

0xc6e00bf3, 0xd5a79147, 0x06ca6351, 0x14292967,

0x27b70a85, 0x2e1b2138, 0x4d2c6dfc, 0x53380d13,

0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85,

0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3,

0xd192e819, 0xd6990624, 0xf40e3585, 0x106aa070,

0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5,

0x391c0cb3, 0x4ed8aa4a, 0x5b9cca4f, 0x682e6ff3,

0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208,

0x90befffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2

]

# Pre-processing: padding the message

message += b'\x80'

while len(message) % 64 != 56:

message += b'\x00'

message += len(message).to\_bytes(8, 'big')

# Process the message in 512-bit blocks

for i in range(0, len(message), 64):

chunk = message[i:i+64]

# Initialize hash values for this chunk

a, b, c, d, e, f, g, h = h0, h1, h2, h3, h4, h5, h6, h7

# Main loop

w = [0] \* 64

for j in range(16):

w[j] = int.from\_bytes(chunk[j\*4:j\*4+4], 'big')

for j in range(16, 64):

s0 = (right\_rotate(w[j-15], 7) ^ right\_rotate(w[j-15], 18) ^ (w[j-15] >> 3))

s1 = (right\_rotate(w[j-2], 17) ^ right\_rotate(w[j-2], 19) ^ (w[j-2] >> 10))

w[j] = (w[j-16] + s0 + w[j-7] + s1) % (1 << 32)

for j in range(64):

s1 = (right\_rotate(e, 6) ^ right\_rotate(e, 11) ^ right\_rotate(e, 25))

ch = (e & f) ^ (~e & g)

temp1 = (h + s1 + ch + k[j] + w[j]) % (1 << 32)

s0 = (right\_rotate(a, 2) ^ right\_rotate(a, 13) ^ right\_rotate(a, 22))

maj = (a & b) ^ (a & c) ^ (b & c)

temp2 = (s0 + maj) % (1 << 32)

h = g

g = f

f = e

e = (d + temp1) % (1 << 32)

d = c

c = b

b = a

a = (temp1 + temp2) % (1 << 32)

# Update hash values for this chunk

h0 = (h0 + a) % (1 << 32)

h1 = (h1 + b) % (1 << 32)

h2 = (h2 + c) % (1 << 32)

h3 = (h3 + d) % (1 << 32)

h4 = (h4 + e) % (1 << 32)

h5 = (h5 + f) % (1 << 32)

h6 = (h6 + g) % (1 << 32)

h7 = (h7 + h) % (1 << 32)

# Produce the final hash value

hash\_value = (h0 << 224) | (h1 << 192) | (h2 << 160) | (h3 << 128) | (h4 << 96) | (h5 << 64) | (h6 << 32) | h7

return hash\_value.to\_bytes(32, 'big')

def right\_rotate(x, n):

return ((x >> n) | (x << (32 - n))) & 0xFFFFFFFF

# Test the SHA-256 implementation

message = b'Test message'

hash\_value = sha256(message)

print("SHA