***SAVEETHA SCHOOL OF ENGINEERING***

***SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCE***

**EXP NO 13 : Calculate the message digest of the text by implementing SHA-512 hashing technique.**

**AIM**

To Calculate the message digest of the text by implementing SHA-512 hashing technique.

**PROCEDURE**

* initialize five random strings of hex characters
* The message is then [padded](https://brilliant.org/wiki/padding/) by appending a 1, followed by enough 0s until the message is 448 bits.
* begin the 80 iterations,I.
* when all the chunks have been processed, the message digest is represented as the 160-bit string comprised of the **OR** logical operator, ∨∨, of the 5 hashed values

**PROGRAM**

#include <stdio.h>

#include <stdint.h>

#include <string.h>

#define BLOCK\_SIZE 128

#define OUTPUT\_SIZE 64

#define ROTR(x, n) (((x) >> (n)) | ((x) << (64 - (n))))

#define CH(x, y, z) (((x) & (y)) ^ (~(x) & (z)))

#define MAJ(x, y, z) (((x) & (y)) ^ ((x) & (z)) ^ ((y) & (z)))

#define SIGMA0(x) (ROTR(x, 28) ^ ROTR(x, 34) ^ ROTR(x, 39))

#define SIGMA1(x) (ROTR(x, 14) ^ ROTR(x, 18) ^ ROTR(x, 41))

#define sigma0(x) (ROTR(x, 1) ^ ROTR(x, 8) ^ ((x) >> 7))

#define sigma1(x) (ROTR(x, 19) ^ ROTR(x, 61) ^ ((x) >> 6))

const uint64\_t H[] = { 0x6a09e667f3bcc908, 0xbb67ae8584caa73b, 0x3c6ef372fe94f82b, 0xa54ff53a5f1d36f1,

0x510e527fade682d1, 0x9b05688c2b3e6c1f, 0x1f83d9abfb41bd6b, 0x5be0cd19137e2179 };

const uint64\_t K[] = { 0x428a2f98d728ae22, 0x7137449123ef65cd, 0xb5c0fbcfec4d3b2f, 0xe9b5dba58189dbbc,

/\*... shortened for brevity ... \*/ };

typedef struct {

uint64\_t words[80];

} sha512\_schedule;

typedef struct {

uint64\_t hash[8];

uint64\_t total\_len;

uint64\_t len;

unsigned char buffer[BLOCK\_SIZE];

} sha512\_ctx;

void sha512\_transform(sha512\_ctx \*ctx);

void sha512\_init(sha512\_ctx \*ctx);

void sha512\_update(sha512\_ctx \*ctx, const unsigned char \*message, uint64\_t len);

void sha512\_final(sha512\_ctx \*ctx, unsigned char \*digest);

void sha512\_transform(sha512\_ctx \*ctx) {

uint64\_t W[80], temp1, temp2, A, B, C, D, E, F, G, H;

int t;

for (t = 0; t < 16; t++)

W[t] = ((uint64\_t)ctx->buffer[t \* 8 + 0] << 56) | ((uint64\_t)ctx->buffer[t \* 8 + 1] << 48) |

((uint64\_t)ctx->buffer[t \* 8 + 2] << 40) | ((uint64\_t)ctx->buffer[t \* 8 + 3] << 32) |

((uint64\_t)ctx->buffer[t \* 8 + 4] << 24) | ((uint64\_t)ctx->buffer[t \* 8 + 5] << 16) |

((uint64\_t)ctx->buffer[t \* 8 + 6] << 8) | ((uint64\_t)ctx->buffer[t \* 8 + 7]);

for (t = 16; t < 80; t++)

W[t] = sigma1(W[t - 2]) + W[t - 7] + sigma0(W[t - 15]) + W[t - 16];

A = ctx->hash[0]; B = ctx->hash[1]; C = ctx->hash[2]; D = ctx->hash[3];

E = ctx->hash[4]; F = ctx->hash[5]; G = ctx->hash[6]; H = ctx->hash[7];

for (t = 0; t < 80; t++) {

temp1 = H + SIGMA1(E) + CH(E, F, G) + K[t] + W[t];

temp2 = SIGMA0(A) + MAJ(A, B, C);

H = G; G = F; F = E; E = D + temp1; D = C; C = B; B = A; A = temp1 + temp2;

}

ctx->hash[0] += A; ctx->hash[1] += B; ctx->hash[2] += C; ctx->hash[3] += D;

ctx->hash[4] += E; ctx->hash[5] += F; ctx->hash[6] += G; ctx->hash[7] += H;

}

void sha512\_init(sha512\_ctx \*ctx) {

memcpy(ctx->hash, H, sizeof(H));

ctx->len = 0; ctx->total\_len = 0;

}

void sha512\_update(sha512\_ctx \*ctx, const unsigned char \*message, uint64\_t len) {

uint64\_t i;

for (i = 0; i < len; ++i) {

ctx->buffer[ctx->len++] = message[i];

if (ctx->len == BLOCK\_SIZE) {

sha512\_transform(ctx);

ctx->total\_len += BLOCK\_SIZE \* 8; ctx->len = 0;

}

}

}

void sha512\_final(sha512\_ctx \*ctx, unsigned char \*digest) {

int i = ctx->len;

if (ctx->len < BLOCK\_SIZE - 16) {

ctx->buffer[i++] = 0x80; memset(ctx->buffer + i, 0, BLOCK\_SIZE - 16 - i);

} else {

ctx->buffer[i++] = 0x80;

while (i < BLOCK\_SIZE - 16) ctx->buffer[i++] = 0x00;

}

ctx->total\_len += ctx->len \* 8;

for (i = 0; i < 8; ++i) {

digest[i \* 8 + 0] = (ctx->hash[i] >> 56) & 0xFF; digest[i \* 8 + 1] = (ctx->hash[i] >> 48) & 0xFF;

digest[i \* 8 + 2] = (ctx->hash[i] >> 40) & 0xFF; digest[i \* 8 + 3] = (ctx->hash[i] >> 32) & 0xFF;

digest[i \* 8 + 4] = (ctx->hash[i] >> 24) & 0xFF; digest[i \* 8 + 5] = (ctx->hash[i] >> 16) & 0xFF;

digest[i \* 8 + 6] = (ctx->hash[i] >> 8) & 0xFF; digest[i \* 8 + 7] = ctx->hash[i] & 0xFF;

}

}

int main() {

const char \*message = "Hello, world!";

unsigned char digest[OUTPUT\_SIZE];

sha512\_ctx ctx;

sha512\_init(&ctx);

sha512\_update(&ctx, (const unsigned char \*)message, strlen(message));

sha512\_final(&ctx, digest);

printf("Message: %s\nSHA-512 Digest: ", message);

for (int i = 0; i < OUTPUT\_SIZE; ++i) printf("%02x", digest[i]);

printf("\n");

return 0;

}

**OUTPUT**

