Assignment 7(SHA 256 algorithm):

Cpp code

```
#include <iostream>
#include <cstring>
using namespace std;

#define uchar unsigned char
#define uint unsigned int
#define DBL_INT_ADD(a, b, c) if(a > 0xffffffff - (c)) ++b; a += c;
#define ROTLEFT(a, b) (((a) << (b)) | ((a) >> (32 - (b))))
#define ROTRIGHT(a, b) (((a) >> (b)) | ((a) << (32 - (b))))
#define CH(x, y, z) (((x) & (y)) ^ (~(x) & (z)))
#define MAJ(x, y, z) (((x) & (y)) ^ ((x) & (z)) ^ ((y) & (z)))
#define EP0(x) (ROTRIGHT(x, 2) ^ ROTRIGHT(x, 13) ^ ROTRIGHT(x, 22))
#define EP1(x) (ROTRIGHT(x, 6) ^ ROTRIGHT(x, 11) ^ ROTRIGHT(x, 25))
#define SIG0(x) (ROTRIGHT(x, 7) ^ ROTRIGHT(x, 18) ^ ((x) >> 3))
#define SIG1(x) (ROTRIGHT(x, 17) ^ ROTRIGHT(x, 19) ^ ((x) >> 10))

typedef struct
{
    uchar data[64];
    uint datalen;
    uint bitlen[2];
    uint state[8];
} SHA256_CONTROL;

uint k[64] = {
    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
};

void SHA256Transform(SHA256_CONTROL *pointer_userDef, uchar data[])
{
    uint h1, h2, h3, h4, h5, h6, h7, h8, i, j, t1, t2, m[64];
    for (i = 0, j = 0; i < 16; ++i, j += 4)
    {
        m[i] = (data[j] << 24) | (data[j + 1] << 16) | (data[j + 2] <<
8) | (data[j + 3]);
    }
    for (; i < 64; ++i)
    {
        m[i] = SIG1(m[i - 2]) + m[i - 7] + SIG0(m[i - 15]) + m[i - 16];
    }
    h1 = pointer_userDef->state[0];
    h2 = pointer_userDef->state[1];
    h3 = pointer_userDef->state[2];
    h4 = pointer_userDef->state[3];
    h5 = pointer_userDef->state[4];
    h6 = pointer_userDef->state[5];
    h7 = pointer_userDef->state[6];
    h8 = pointer_userDef->state[7];

    for (i = 0; i < 64; ++i)
    {
        t1 = h8 + EP1(h5) + CH(h5, h6, h7) + k[i] + m[i];
        t2 = EP0(h1) + MAJ(h1, h2, h3);
        h8 = h7;
        h7 = h6;
        h6 = h5;
        h5 = h4 + t1;
        h4 = h3;
        h3 = h2;
        h2 = h1;
        h1 = t1 + t2;
    }

    pointer_userDef->state[0] += h1;
    pointer_userDef->state[1] += h2;
    pointer_userDef->state[2] += h3;
    pointer_userDef->state[3] += h4;
    pointer_userDef->state[4] += h5;
    pointer_userDef->state[5] += h6;
    pointer_userDef->state[6] += h7;
    pointer_userDef->state[7] += h8;
}

int main()
{
    char data[] = "hi i am robot this is a encypted text 0198768";
    int strLen = strlen(data);
    SHA256_CONTROL pointer_userDef;
    SHA256_CONTROL *pointer = &pointer_userDef;
    unsigned char hash[32];
    string hashStr = "";

    pointer->datalen = 0;
    pointer->bitlen[0] = 0;
    pointer->bitlen[1] = 0;
    pointer->state[0] = 0x6a09e667;
    pointer->state[1] = 0xbb67ae85;
    pointer->state[2] = 0x3c6ef372;
    pointer->state[3] = 0xa54ff53a;
    pointer->state[4] = 0x510e527f;
    pointer->state[5] = 0x9b05688c;
    pointer->state[6] = 0x1f83d9ab;
    pointer->state[7] = 0x5be0cd19;

    for (uint i = 0; i < strLen; ++i)
    {
        pointer->data[pointer->datalen] = data[i];
        pointer->datalen++;
        if (pointer->datalen == 64)
        {
            SHA256Transform(pointer, pointer->data);
            DBL_INT_ADD(pointer->bitlen[0], pointer->bitlen[1], 512);
            pointer->datalen = 0;
        }
    }

    uint i = pointer->datalen;
    if (pointer->datalen < 56)
    {
        pointer->data[i++] = 0x80;
        while (i < 56)
        {
            pointer->data[i++] = 0x00;
        }
    }
    else
    {
        pointer->data[i++] = 0x80;
        while (i < 64)
        {
            pointer->data[i++] = 0x00;
        }
        SHA256Transform(pointer, pointer->data);
        memset(pointer->data, 0, 56);
    }
    DBL_INT_ADD(pointer->bitlen[0], pointer->bitlen[1],
pointer->datalen * 8);

    pointer->data[63] = pointer->bitlen[0];
    pointer->data[62] = pointer->bitlen[0] >> 8;
    pointer->data[61] = pointer->bitlen[0] >> 16;
    pointer->data[60] = pointer->bitlen[0] >> 24;
    pointer->data[59] = pointer->bitlen[1];
    pointer->data[58] = pointer->bitlen[1] >> 8;
    pointer->data[57] = pointer->bitlen[1] >> 16;
    pointer->data[56] = pointer->bitlen[1] >> 24;

    SHA256Transform(pointer, pointer->data);
    for (i = 0; i < 4; ++i)
    {
        hash[i] = (pointer->state[0] >> (24 - i * 8)) & 0x000000ff;
        hash[i + 4] = (pointer->state[1] >> (24 - i * 8)) & 0x000000ff;
        hash[i + 8] = (pointer->state[2] >> (24 - i * 8)) & 0x000000ff;
        hash[i + 12] = (pointer->state[3] >> (24 - i * 8)) &
0x000000ff;
        hash[i + 16] = (pointer->state[4] >> (24 - i * 8)) &
0x000000ff;
        hash[i + 20] = (pointer->state[5] >> (24 - i * 8)) &
0x000000ff;
        hash[i + 24] = (pointer->state[6] >> (24 - i * 8)) &
0x000000ff;
        hash[i + 28] = (pointer->state[7] >> (24 - i * 8)) &
0x000000ff;
```

```
    }

    char s[3];
    for (int i = 0; i < 32; i++)
    {
        sprintf(s, "%02x", hash[i]);
        hashStr += s;
    }
    cout << hashStr;
    return 0;
};
```

- Test cases passed
- Completed on 22/4/23

Q/A:

1. How long did you spend on this assignment?
    a. 1day
2. Based on your effort, what letter grade would you say you earned?
    a. On a scale of 1 to 10. I would grade this as 10/10.
3. Based on your solution, what letter grade would you say you earned?
    a. On a scale of 1 to 10. I would grade this as 9/10.
4. Provide a summary of what doesn't work in your solution, along with an explanation of how you attempted to solve the problem and where you feel you struggled?
    a. My solution is based on sha256 algorithm explanation in wikipedia. I felt quite hard at implementing rotation operations on data.