CRYPTOGRAPHY LABORATORY FILE
CS-511

Submitted to:
Dr. Samayveer Singh
Assistant Professor
Department of Computer Science

Submitted by:
Shashi Shekhar Azad
Roll No.: 23203029
M. Tech. CSE 1st Semester

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
DR. B. R. AMBEDKAR NATIONAL INSTITUTE OF TECHNOLOGY
JALANDHAR

Assignment 11

Write a program to implement the MD5 Hash function to validate data integrity.

Code:

```
#include <bits/stdc++.h>
using namespace std;
#define roundF(x, y, z) (((x) & (y)) | ((~x) & (z)))
#define roundG(x, y, z) (((x) & (z)) | ((y) & (~z)))
#define roundH(x, y, z) ((x) ^ (y) ^ (z))
#define roundI(x, y, z) ((y) ^ ((x) | (~z)))
#define leftRotate(x, n) (((x) << (n)) | ((x) >> (32 - (n))))
string littleEndian64Bits(string str)
{
    string b1, b2, b3, b4, b5, b6, b7, b8;
    for (int i = 0; i < 64; i++)
    {
        if ((i >= 0) & (i < 8))
            b1 += str[i];
        if ((i >= 8) & (i < 16))
            b2 += str[i];
        if ((i >= 16) & (i < 24))
            b3 += str[i];
        if ((i >= 24) & (i < 32))
            b4 += str[i];
        if ((i >= 32) & (i < 40))
            b5 += str[i];
        if ((i >= 40) & (i < 48))
            b6 += str[i];
        if ((i >= 48) & (i < 56))
            b7 += str[i];
        if ((i >= 56) & (i < 64))
            b8 += str[i];
    }
    return b8 + b7 + b6 + b5 + b4 + b3 + b2 + b1;
}
string stringToBinary(string msg)
{
    string binary;
    for (size_t i = 0; i < msg.size(); ++i)
        binary += bitset<8>(msg.c_str()[i]).to_string();
    return binary;
}
string numberToBinary(unsigned number, int size)
{
    string binary;
    while (number != 0)
    {
        binary = (number % 2 == 0 ? "0" : "1") + binary;
        number /= 2;
    }
    if (size > -1)
    {
        if (size > binary.length())
        {
            string temp;
            for (int i = (int)binary.length(); i < size; i++)
            {
                temp += '0';
            }
            binary = temp + binary;
        }
    }
    return binary;
}
string appendTo512Bits(string msg)
{
    string result = msg;
    int string_size = (int)msg.length();
    result += '1';
    int fullSize;
    if ((string_size % 512) < 448)
        fullSize = 512 * (string_size / 512) + 448 - string_size - 1;
    else
        fullSize = 512 * (string_size / 512) + 511 - string_size + 448;
    for (int i = 0; i < fullSize; i++)
        result += '0';
    string string_size_binary = numberToBinary(string_size, -1);
    string result_lenght;
    for (int i = 0; i < (64 - string_size_binary.size()); i++)
```

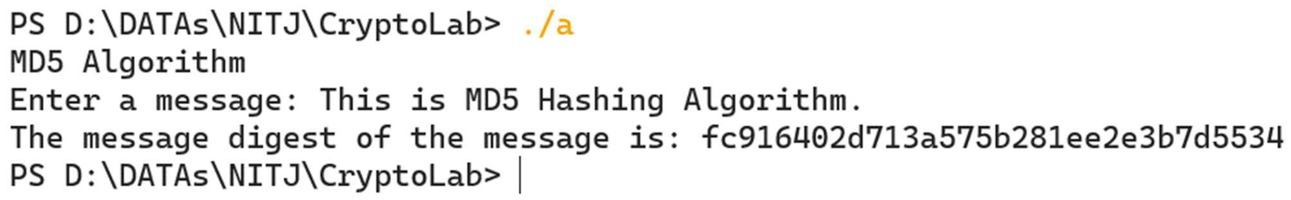
```
        result_lenght += '0';
    result_lenght += string_size_binary;
    result_lenght = littleEndian64Bits(result_lenght);
    return result + result_lenght;
}
string reverse(string to_reverse)
{
    string result;
    for (int i = (int)to_reverse.length() - 1; i >= 0; i--)
        result += to_reverse[i];
    return result;
}
string littleEndian32Bits(string str)
{
    string b1, b2, b3, b4;
    for (int i = 0; i < 32; i++)
    {
        if ((i >= 0) & (i < 8))
            b1 += str[i];
        if ((i >= 8) & (i < 16))
            b2 += str[i];
        if ((i >= 16) & (i < 24))
            b3 += str[i];
        if ((i >= 24) & (i < 32))
            b4 += str[i];
    }

    return b4 + b3 + b2 + b1;
}
unsigned binaryStringToDecimal(string a)
{
    unsigned num = 0;
    bool neg = false;
    if (a.at(0) == '1')
    {
        neg = true;
        for (int x = (int)a.length() - 1; x >= 0; x--)
        {
            if (a.at(x) == '1')
```

```
                a.at(x) = '0';
            else
                a.at(x) = '1';
        }
        a.at(a.length() - 1) += 1;
        for (int x = (int)a.length() - 1; x >= 0; x--)
        {
            if (a.at(x) == '2')
            {
                if (x - 1 >= 0)
                {
                    if (a.at(x - 1) == '1')
                        a.at(x - 1) = '2';
                    if (a.at(x - 1) == '0')
                        a.at(x - 1) = '1';
                    a.at(x) = '0';
                }
            }
            else if (a.at(x) == '3')
            {
                if (x - 1 >= 0)
                    a.at(x - 1) += '2';
                a.at(x) = '1';
            }
        }
        if (a.at(0) == '2')
            a.at(0) = '0';
        else if (a.at(0) == '3')
            a.at(0) = '1';
    }
    for (int x = (int)a.length() - 1; x >= 0; x--)
    {
        if (a.at(x) == '1')
            num += pow(2.0, a.length() - x - 1);
    }
    if (neg)
        num = num * -1;
    return num;
}
```

```
int main(int argc, const char *argv[])
{
    int n = 0;
    string message;
    string bits_message;
    unsigned k[64];
    unsigned s[64] = {
        7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22,
        5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20,
        4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23,
        6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21};
    for (int i = 0; i < 64; i++)
    {
        k[i] = floor(pow(2, 32) * abs(sin(i + 1)));
    }
    unsigned h0 = 0x67452301;
    unsigned h1 = 0xefcdab89;
    unsigned h2 = 0x98badcfe;
    unsigned h3 = 0x10325476;
    cout << "MD5 Algorithm" << endl;
    cout << "Enter a message: ";
    getline(cin, message);
    bits_message = stringToBinary(message);
    bits_message = appendTo512Bits(bits_message);
    n = (int)bits_message.length() / 32;
    for (int i = 0; i <= (n / 16) - 1; i++)
    {
        string m[16];
        for (int j = 0; j < 16; j++)
        {
            m[j] = littleEndian32Bits(bits_message.substr(j * 32, 32));
        }
        unsigned a = h0;
        unsigned b = h1;
        unsigned c = h2;
        unsigned d = h3;
        unsigned f = 0;
        int g = 0;
        for (int i = 0; i < 64; i++)
    {
        if (i >= 0 && i < 16)
        {
            f = roundF(b, c, d);
            g = i;
        }
        else if (i >= 16 && i < 32)
        {
            f = roundG(b, c, d);
            g = (5 * i + 1) % 16;
        }
        else if (i >= 32 && i < 48)
        {
            f = roundH(b, c, d);
            g = (3 * i + 5) % 16;
        }
        else if (i >= 48 && i < 64)
        {
            f = roundI(b, c, d);
            g = (7 * i) % 16;
        }
        f = f + a + k[i] + binaryStringToDecimal(m[g]);
        a = d;
        d = c;
        c = b;
        b = b + leftRotate(f, s[i]);
    }
    h0 = h0 + a;
    h1 = h1 + b;
    h2 = h2 + c;
    h3 = h3 + d;
}
string hArray[4] = {
    littleEndian32Bits(numberToBinary(h0, 32)),
    littleEndian32Bits(numberToBinary(h1, 32)),
    littleEndian32Bits(numberToBinary(h2, 32)),
    littleEndian32Bits(numberToBinary(h3, 32))};
cout << "The message digest of the message is: ";
for (int h = 0; h < 4; h++)
```

```
    {
        string bin(hArray[h]);
        int result = 0;
        for (size_t count = 0; count < bin.length(); ++count)
        {
            result *= 2;
            result += bin[count] == '1' ? 1 : 0;
        }
        cout << hex << setw(8) << setfill('0') << result;
    }
    return 0;
}
```

Output:



```
PS D:\DATAs\NITJ\CryptoLab> ./a
MD5 Algorithm
Enter a message: This is MD5 Hashing Algorithm.
The message digest of the message is: fc916402d713a575b281ee2e3b7d5534
PS D:\DATAs\NITJ\CryptoLab>
```