实验五 Hash函数MD5

学号：　姓名：　专业：

一、实验目的

通过实际编程了解MD5算法的过程，加深对Hash函数的认识。

二、实验内容

1. 算法分析：请参照教材内容，分析MD5算法实现的每一步原理。
2. 算法实现：利用Visual C＋＋语言，自己编写MD5的实现代码，并检验代码实现的正确性。
3. 雪崩效应检验：尝试对一个长字符串进行Hash运算，并获得其运算结果。对该字符串进行轻微的改动，比如增加一个空格或标点，比较Hash结果值的改变位数。进行8次这样的测试。

三. MD5算法

1. 流程图

屏幕截图 2022-12-29 134951

2. 代码

- **逻辑函数**

```
inline unsigned int F(unsigned int X, unsigned int Y, unsigned int Z)
{
    return (X & Y) | ((~X) & Z);
}
inline unsigned int G(unsigned int X, unsigned int Y, unsigned int Z)
{
    return (X & Z) | (Y & (~Z));
}
inline unsigned int H(unsigned int X, unsigned int Y, unsigned int Z)
{
    return X ^ Y ^ Z;
}
inline unsigned int I(unsigned int X, unsigned int Y, unsigned int Z)
{
    return Y ^ (X | (~Z));
}
```

- ### 循环左移

```
void ROL(unsigned int& s, unsigned short cx)//循环左移
{
    if (cx > 32)cx %= 32;
    s = (s << cx) | (s >> (32 - cx));
    return;
}7
```

- ### 循环计算函数

```
//循环左移-位数表
const unsigned int rolarray[4][4] = {
        { 7, 12, 17, 22 },
        { 5, 9, 14, 20 },
        { 4, 11, 16, 23 },
        { 6, 10, 15, 21 }
};

//数据坐标表(对消息中字的顺序做变换, 得到的置换表)
const unsigned short mN[4][16] = {
    { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 },
    { 1, 6, 11, 0, 5, 10, 15, 4, 9, 14, 3, 8, 13, 2, 7, 12 },
    { 5, 8, 11, 14, 1, 4, 7, 10, 13, 0, 3, 6, 9, 12, 15, 2 },
    { 0, 7, 14, 5, 12, 3, 10, 1, 8, 15, 6, 13, 4, 11, 2, 9 }
};

//常数表
const unsigned int T[64] = {
0xD76AA478,0xE8C7B756,0x242070DB,0xC1BDCEEE,0xF57C0FAF,0x4787C62A,0xA8304613,0xFD469501,
0x698098D8,0x8B44F7AF,0xFFFF5BB1,0x895CD7BE,0x6B901122,0xFD987193,0xA679438E,0x49B40821,
0xF61E2562,0xC040B340,0x265E5A51,0xE9B6C7AA,0xD62F105D,0x02441453,0xD8A1E681,0xE7D3FBC8,
0x21E1CDE6,0xC33707D6,0xF4D50D87,0x455A14ED,0xA9E3E905,0xFCEFA3F8,0x676F02D9,0x8D2A4C8A,
0xFFFA3942,0x8771F681,0x6D9D6122,0xFDE5380C,0xA4BEEA44,0x4BDECFA9,0xF6BB4B60,0xBEBFBC70,
0x289B7EC6,0xEAA127FA,0xD4EF3085,0x04881D05,0xD9D4D039,0xE6DB99E5,0x1FA27CF8,0xC4AC5665,
0xF4292244,0x432AFF97,0xAB9423A7,0xFC93A039,0x655B59C3,0x8F0CCC92,0xFFEFF47D,0x85845DD1,
0x6FA87E4F,0xFE2CE6E0,0xA3014314,0x4E0811A1,0xF7537E82,0xBD3AF235,0x2AD7D2BB,0xEB86D391
};

//MD5循环计算函数，lGroup 存储ABCD的值，M 数据分组（16组32位数指针）
```

```
void AccLoop(unsigned short rounds, unsigned int* lGroup, void* M)
{
    unsigned int tmpi = 0; //T是累加器
    unsigned int A, B, C, D;
    typedef unsigned int(*clac)(unsigned int X, unsigned int Y, unsigned int Z); //定义函数类型

    const unsigned int* pM = static_cast<unsigned int*>(M);//转换类型为32位的Uint
    clac clacArr[4] = { F, G, H, I }; //定义并初始化计算函数指针数组

    /*一轮循环开始（16组->16次）*/
    for (short i = 0; i < 16; ++i)
    {
        A = lGroup[0];
        B = lGroup[1];
        C = lGroup[2];
        D = lGroup[3];

        //计算A+F(B,C,D)+M[i]+T
        tmpi = A + clacArr[rounds](B, C, D) + pM[(mN[rounds][i])] + T[rounds * 16 + i];
        ROL(tmpi, rolarray[rounds][i % 4]);//循环左移

        //给缓冲区赋值
        lGroup[0] = D;
        lGroup[1] = tmpi + B;
        lGroup[2] = B;
        lGroup[3] = C;
    }
    return;
}
```

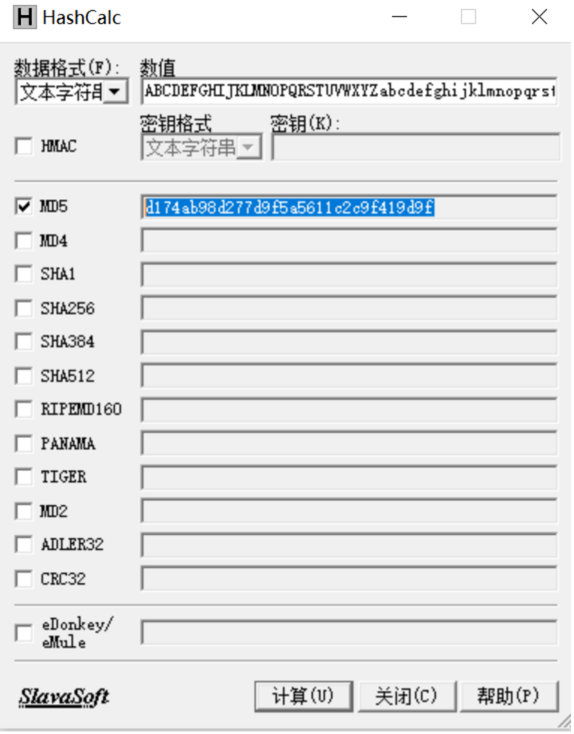
- **MD5整体框架**

```
unsigned int* MD5(const char* mStr) {
    unsigned int mLen = strlen(mStr); //计算字符串长度

    unsigned int FillSize;
    if ((mLen * 8) % 512 <= 448) {
        FillSize = 448 - ((mLen * 8) % 512); //计算需填充的bit数
    }
    else {
        FillSize = 512 + 448 - ((mLen * 8) % 512);
    }

    if (FillSize == 0) {
        FillSize = 512;
    }

    unsigned int FSbyte = FillSize / 8; //以字节表示的填充数
    unsigned int BuffLen = mLen + 8 + FSbyte; //填充后的长度
    unsigned char* md5Buff = new unsigned char[BuffLen]; //分配缓冲区
    memcpy(md5Buff, mStr, mLen); //复制字符串到缓冲区

    /*数据填充开始*/
    md5Buff[mLen] = 0x80; //第一个bit填充1
    ZeroMemory(&md5Buff[mLen + 1], FSbyte - 1); //其它bit填充0
    unsigned long long lenBit = mLen * 8ULL; //计算字符串长度，准备填充
    memcpy(&md5Buff[mLen + FSbyte], &lenBit, 8); //填充消息长度
    /*数据填充结束*/

    /*运算开始*/
    unsigned int LoopNumber = BuffLen / 64; //以16个字为一分组，计算分组数量
    unsigned int A = 0x67452301, B = 0x0EFCDAB89, C = 0x98BADCFE, D = 0x10325476;//初始4个种子，小端类型

    unsigned int* lGroup = new unsigned int[4]{ A, B, C, D }; //种子副本数组,并作为返回值返回
    for (unsigned int Bcount = 0; Bcount < LoopNumber; Bcount++) //分组大循环开始
    {
        /*进入4次计算的小循环*/
        for (unsigned short Lcount = 0; Lcount < 4; Lcount++)
        {
            AccLoop(Lcount, lGroup, &md5Buff[Bcount * 64]);
        }
        /*数据相加作为下一轮的种子或者最终输出*/
        A = (lGroup[0] += A);
        B = (lGroup[1] += B);
        C = (lGroup[2] += C);
        D = (lGroup[3] += D);
    }

    delete[] md5Buff; //清除内存并返回
    return lGroup;
}
```

3. 程序验证

采用提供的测试样例中长字符串：ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789

采用MD5算法加密后结果为： 0xd1, 0x74, 0xab, 0x98, 0xd2, 0x77, 0xd9, 0xf5, 0xa5, 0x61, 0x1c, 0x2c, 0x9f, 0x41, 0x9d, 0x9f



HashCalc

数据格式(F): 文本字符串 数值 ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrst

□ HMAC 密钥格式 文本字符串 密钥(K):

☑ MD5 d174ab98d277d9f5a5611c2c9f419d9f

□ MD4

□ SHA1

□ SHA256

□ SHA384

□ SHA512

□ RIPEMD160

□ PANAMA

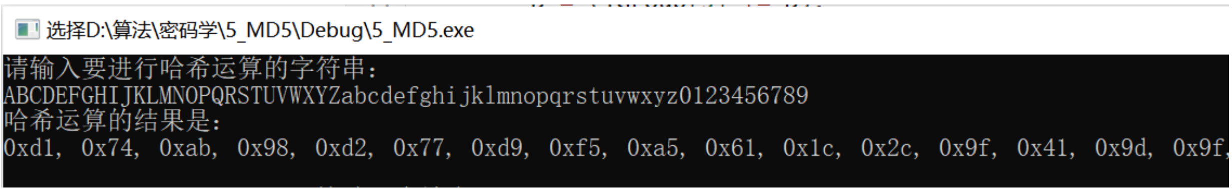
□ TIGER

□ MD2

□ ADLER32

□ CRC32

□ eDonkey/eMule



选择D:\算法\密码学\5_MD5\Debug\5_MD5.exe

```
请输入要进行哈希运算的字符串:
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
哈希运算的结果是:
0xd1, 0x74, 0xab, 0x98, 0xd2, 0x77, 0xd9, 0xf5, 0xa5, 0x61, 0x1c, 0x2c, 0x9f, 0x41, 0x9d, 0x9f,
```

四. 雪崩效应

1. 方法

对字符串ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789进行8次雪崩实验，每次随机改变一个字符为空格或标点或字母或数字，统计每次Hash结果值的二进制改变位数，最终计算8次雪崩测试平均二进制Hash值改变位数为63位

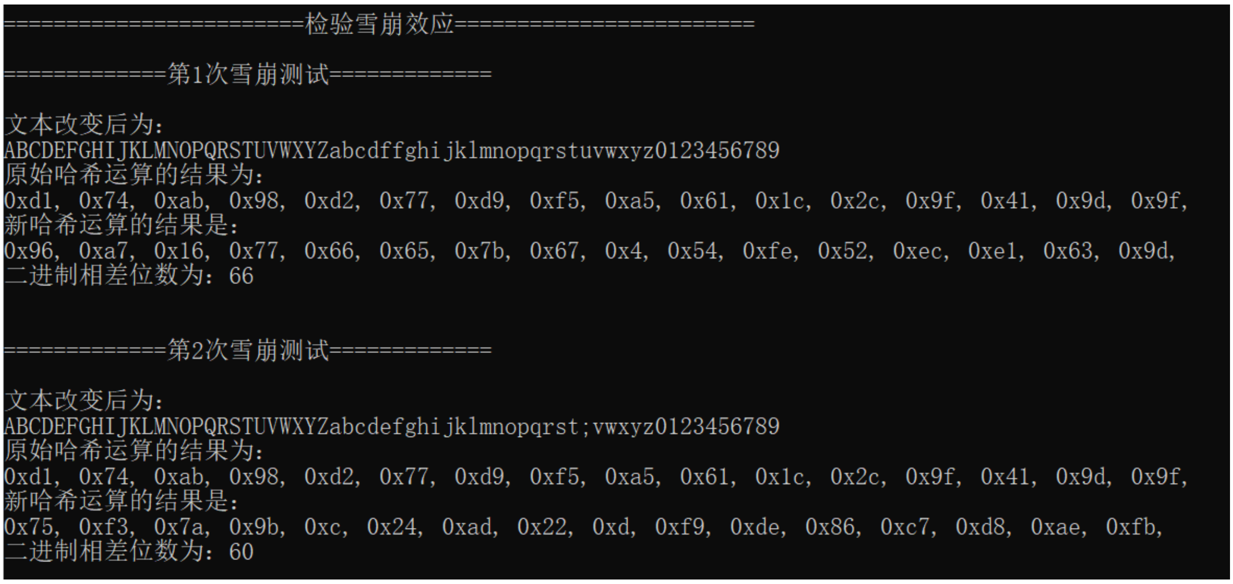
2. 代码

```
// 十进制转二进制
void dex_bin(int num, int bin[])
{
    int length = 0, j = 0;
    do
    {
        bin[j] = num % 2;
        num = num / 2;
        j++;
        length++;
    } while (num != 0);
    while (length != 8)
        bin[length++] = 0;
```

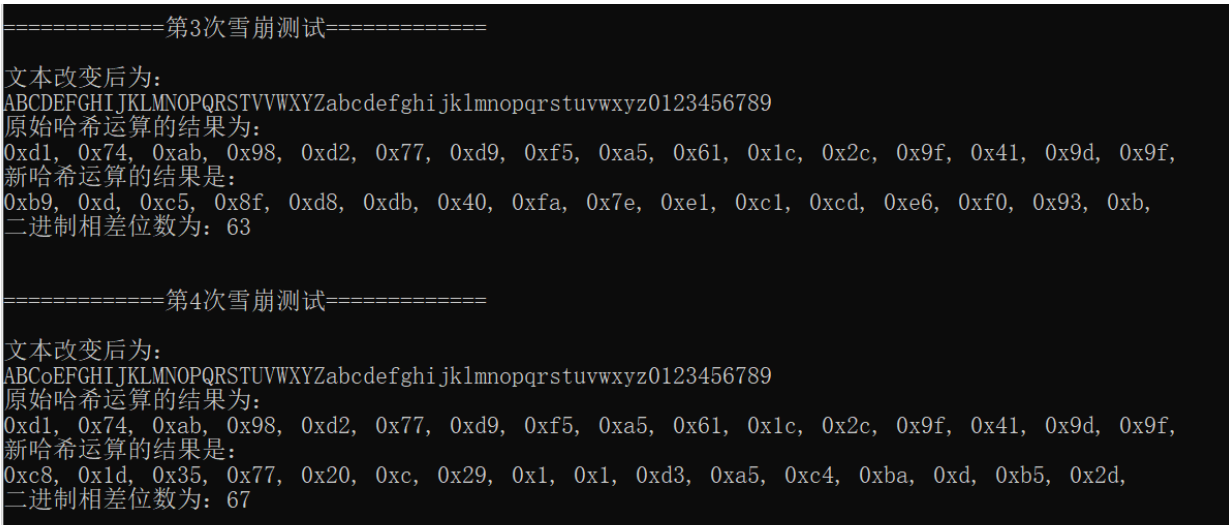
```
}

// 计算二进制不同位数
int cal(int a,int b)
{
    int bin1[8], bin2[8], total = 0;
    dex_bin(a, bin1);
    dex_bin(b, bin2);
    for (int j = 0; j <= 8; j++)
        if (bin1[j] != bin2[j])
            total++;
    return total;
}
int main()
{
    while (1) {
        string s;
        cout << "请输入要进行哈希运算的字符串：" << endl;
        getline(cin, s);
        cout << "哈希运算的结果是：" << endl;
        unsigned int* lGroup = MD5(s.c_str());
        datas_t* datas = (datas_t*)lGroup;

        for (int i = 0; i < 16; i++)
            cout << "0x" << hex << (unsigned int)(unsigned char)datas->data[i] << ", ";
        cout << endl << endl;

        cout << "========================检验雪崩效应========================" << endl << endl;
        // 改变文本
        int len = s.length();
        string text[8];
        srand((unsigned int)time(NULL));
        for (int i = 0; i < 8; i++) {
            int index = rand() % len;
            int randomNum = rand() % 91+34;
            text[i] = s;
            text[i][index] = randomNum;
        }
        int avg = 0;
        for (int i = 0; i < 8; i++)
        {
            cout << "=============第" << i + 1 << "次雪崩测试=============" << endl << endl;;
            cout << "文本改变后为：" << endl;
            cout << text[i] << endl;

            cout << "原始哈希运算的结果为：" << endl;
            for (int j = 0; j < 16; j++)
                cout << "0x" << (hex) << (unsigned int)(unsigned char)datas->data[j] << ", ";
            cout << endl;

            cout << "新哈希运算的结果是：" << endl;
            unsigned int* lGroup = MD5(text[i].c_str());
            datas_t* temp = (datas_t*)lGroup;
            for (int j = 0; j < 16; j++)
```

```
                cout << "0x" << (hex) << (unsigned int)(unsigned char)temp->data[j] << ", ";
            cout << endl;
            int total = 0;
            for (int k = 0; k < 16;k++) {
                int num, ori;
                num = (unsigned int)(unsigned char)temp->data[k];
                ori = (unsigned int)(unsigned char)datas->data[k];
                total += cal(num, ori);
            }
            avg += total;
            cout << "二进制相差位数为：" << dec << total << endl;
            cout << endl << endl;
        }
        cout << "8次雪崩测试二进制平均改变位数为：" << dec << avg / 8 << endl;
    }
}
```

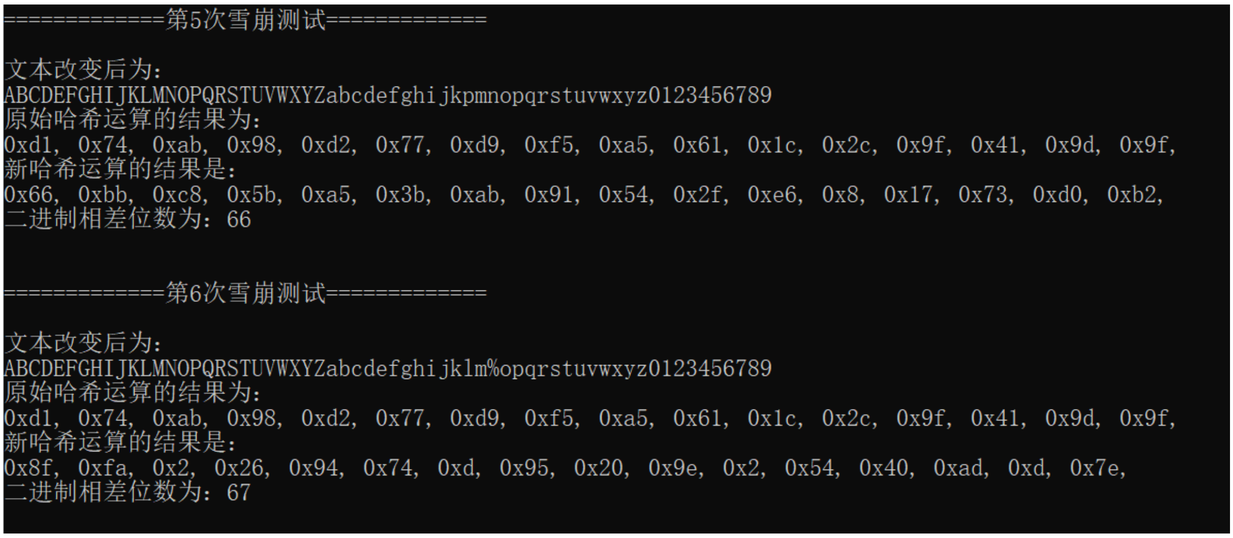
3. 程序运行



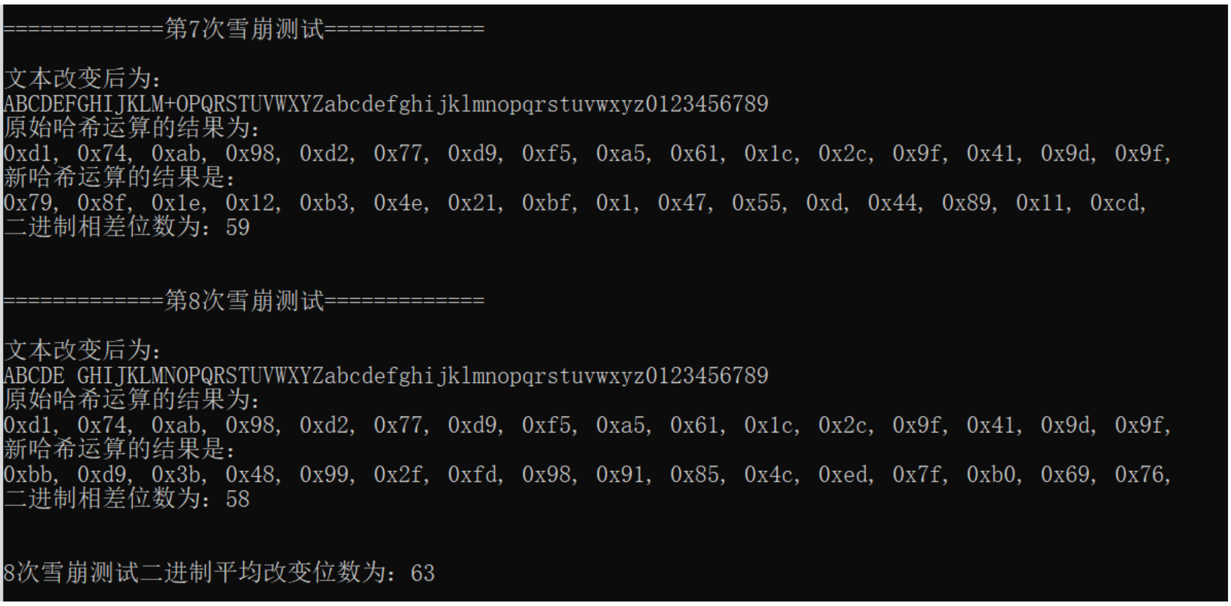
```
========================检验雪崩效应========================

=============第1次雪崩测试=============

文本改变后为：
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdffghijklmnopqrstuvwxyz0123456789
原始哈希运算的结果为：
0xd1，0x74，0xab，0x98，0xd2，0x77，0xd9，0xf5，0xa5，0x61，0x1c，0x2c，0x9f，0x41，0x9d，0x9f，
新哈希运算的结果是：
0x96，0xa7，0x16，0x77，0x66，0x65，0x7b，0x67，0x4，0x54，0xfe，0x52，0xec，0xe1，0x63，0x9d，
二进制相差位数为：66


=============第2次雪崩测试=============

文本改变后为：
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrst;vwxyz0123456789
原始哈希运算的结果为：
0xd1，0x74，0xab，0x98，0xd2，0x77，0xd9，0xf5，0xa5，0x61，0x1c，0x2c，0x9f，0x41，0x9d，0x9f，
新哈希运算的结果是：
0x75，0xf3，0x7a，0x9b，0xc，0x24，0xad，0x22，0xd，0xf9，0xde，0x86，0xc7，0xd8，0xae，0xfb，
二进制相差位数为：60
```



```
=============第3次雪崩测试=============

文本改变后为：
ABCDEFGHIJKLMNOPQRSTVVWXYZabcdefghijklmnopqrstuvwxyz0123456789
原始哈希运算的结果为：
0xd1，0x74，0xab，0x98，0xd2，0x77，0xd9，0xf5，0xa5，0x61，0x1c，0x2c，0x9f，0x41，0x9d，0x9f，
新哈希运算的结果是：
0xb9，0xd，0xc5，0x8f，0xd8，0xdb，0x40，0xfa，0x7e，0xe1，0xc1，0xcd，0xe6，0xf0，0x93，0xb，
二进制相差位数为：63


=============第4次雪崩测试=============

文本改变后为：
ABCoEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
原始哈希运算的结果为：
0xd1，0x74，0xab，0x98，0xd2，0x77，0xd9，0xf5，0xa5，0x61，0x1c，0x2c，0x9f，0x41，0x9d，0x9f，
新哈希运算的结果是：
0xc8，0x1d，0x35，0x77，0x20，0xc，0x29，0x1，0x1，0xd3，0xa5，0xc4，0xba，0xd，0xb5，0x2d，
二进制相差位数为：67
```



```
=============第5次雪崩测试=============

文本改变后为：
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijkpmnopqrstuvwxyz0123456789
原始哈希运算的结果为：
0xd1， 0x74， 0xab， 0x98， 0xd2， 0x77， 0xd9， 0xf5， 0xa5， 0x61， 0x1c， 0x2c， 0x9f， 0x41， 0x9d， 0x9f，
新哈希运算的结果是：
0x66， 0xbb， 0xc8， 0x5b， 0xa5， 0x3b， 0xab， 0x91， 0x54， 0x2f， 0xe6， 0x8， 0x17， 0x73， 0xd0， 0xb2，
二进制相差位数为：66


=============第6次雪崩测试=============

文本改变后为：
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklm%opqrstuvwxyz0123456789
原始哈希运算的结果为：
0xd1， 0x74， 0xab， 0x98， 0xd2， 0x77， 0xd9， 0xf5， 0xa5， 0x61， 0x1c， 0x2c， 0x9f， 0x41， 0x9d， 0x9f，
新哈希运算的结果是：
0x8f， 0xfa， 0x2， 0x26， 0x94， 0x74， 0xd， 0x95， 0x20， 0x9e， 0x2， 0x54， 0x40， 0xad， 0xd， 0x7e，
二进制相差位数为：67
```



```
=============第7次雪崩测试=============

文本改变后为：
ABCDEFGHIJKLM+OPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
原始哈希运算的结果为：
0xd1， 0x74， 0xab， 0x98， 0xd2， 0x77， 0xd9， 0xf5， 0xa5， 0x61， 0x1c， 0x2c， 0x9f， 0x41， 0x9d， 0x9f，
新哈希运算的结果是：
0x79， 0x8f， 0x1e， 0x12， 0xb3， 0x4e， 0x21， 0xbf， 0x1， 0x47， 0x55， 0xd， 0x44， 0x89， 0x11， 0xcd，
二进制相差位数为：59


=============第8次雪崩测试=============

文本改变后为：
ABCDE GHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
原始哈希运算的结果为：
0xd1， 0x74， 0xab， 0x98， 0xd2， 0x77， 0xd9， 0xf5， 0xa5， 0x61， 0x1c， 0x2c， 0x9f， 0x41， 0x9d， 0x9f，
新哈希运算的结果是：
0xbb， 0xd9， 0x3b， 0x48， 0x99， 0x2f， 0xfd， 0x98， 0x91， 0x85， 0x4c， 0xed， 0x7f， 0xb0， 0x69， 0x76，
二进制相差位数为：58


8次雪崩测试二进制平均改变位数为：63
```