```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>

typedef union uwb {
   unsigned w;
   unsigned char b[4];
} WBunion;

typedef unsigned Digest[4];

unsigned f0( unsigned abcd[] ){
   return ( abcd[1] & abcd[2]) | (~abcd[1] & abcd[3]);}

unsigned f1( unsigned abcd[] ){
   return ( abcd[3] & abcd[1]) | (~abcd[3] & abcd[2]);}

unsigned f2( unsigned abcd[] ){
   return  abcd[1] ^ abcd[2] ^ abcd[3];}

unsigned f3( unsigned abcd[] ){
   return abcd[2] ^ (abcd[1] |~ abcd[3]);}

typedef unsigned (*DgstFctn)(unsigned a[]);

unsigned *calcKs( unsigned *k)
{
   double s, pwr;
```

```
    int i;

    pwr = pow( 2, 32);
    for (i=0; i<64; i++) {
        s = fabs(sin(1+i));
        k[i] = (unsigned)( s * pwr );
    }
    return k;
}

// ROtate v Left by amt bits
unsigned rol( unsigned v, short amt )
{
    unsigned  msk1 = (1<<amt) -1;
    return ((v>>(32-amt)) & msk1) | ((v<<amt) & ~msk1);
}

unsigned *md5( const char *msg, int mlen)
{
    static Digest h0 = { 0x67452301, 0xEFCDAB89, 0x98BADCFE, 0x10325476 };
//    static Digest h0 = { 0x01234567, 0x89ABCDEF, 0xFEDCBA98, 0x76543210 };
    static DgstFctn ff[] = { &f0, &f1, &f2, &f3 };
    static short M[] = { 1, 5, 3, 7 };
    static short O[] = { 0, 1, 5, 0 };
    static short rot0[] = { 7,12,17,22};
    static short rot1[] = { 5, 9,14,20};
    static short rot2[] = { 4,11,16,23};
    static short rot3[] = { 6,10,15,21};
    static short *rots[] = {rot0, rot1, rot2, rot3 };
```

```
static unsigned kspace[64];
static unsigned *k;

static Digest h;
Digest abcd;
DgstFctn fctn;
short m, o, g;
unsigned f;
short *rotn;
union {
   unsigned w[16];
   char     b[64];
}mm;
int os = 0;
int grp, grps, q, p;
unsigned char *msg2;

if (k==NULL) k= calcKs(kspace);

for (q=0; q<4; q++) h[q] = h0[q];   // initialize

{
   grps  = 1 + (mlen+8)/64;
   msg2 = malloc( 64*grps);
   memcpy( msg2, msg, mlen);
   msg2[mlen] = (unsigned char)0x80;
   q = mlen + 1;
   while (q < 64*grps){ msg2[q] = 0; q++ ; }
   {
```

```
//          unsigned char t;
            WBunion u;
            u.w = 8*mlen;
//          t = u.b[0]; u.b[0] = u.b[3]; u.b[3] = t;
//          t = u.b[1]; u.b[1] = u.b[2]; u.b[2] = t;
            q -= 8;
            memcpy(msg2+q, &u.w, 4 );
         }
      }

      for (grp=0; grp<grps; grp++)
      {
         memcpy( mm.b, msg2+os, 64);
         for(q=0;q<4;q++) abcd[q] = h[q];
         for (p = 0; p<4; p++) {
            fctn = ff[p];
            rotn = rots[p];
            m = M[p]; o= O[p];
            for (q=0; q<16; q++) {
               g = (m*q + o) % 16;
               f = abcd[1] + rol( abcd[0]+ fctn(abcd) + k[q+16*p] + mm.w[g], rotn[q%4]);

               abcd[0] = abcd[3];
               abcd[3] = abcd[2];
               abcd[2] = abcd[1];
               abcd[1] = f;
            }
         }
         for (p=0; p<4; p++)
```

```
        h[p] += abcd[p];
      os += 64;
   }

   if( msg2 )
      free( msg2 );

   return h;
}

int main( int argc, char *argv[] )
{
   int j,k;
   const char *msg = "The quick brown fox jumps over the lazy dog.";
   unsigned *d = md5(msg, strlen(msg));
   WBunion u;

   printf("= 0x");
   for (j=0;j<4; j++){
      u.w = d[j];
      for (k=0;k<4;k++) printf("%02x",u.b[k]);
   }
   printf("\n");

   return 0;
}
```