```
#include<iostream>
#include<stdio.h>
#include<stdint.h>
#include<string>
#include<stdlib.h>
#include<cmath>
#include<vector>
#include<stdint.h>
#include<bitset>
using namespace std;
uint32_t reverse(uint32_t q);           //Function Declaration
unsigned int bitCount (unsigned int value);
int main()
{

//Number of per round shifts amount
const uint32_t shifts[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};
//Sine value of Integers
const uint32_t keys[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 };

//Variables initialisation
uint32_t a0=0x67452301;     //A
uint32_t b0=0xefcdab89;        //B
uint32_t c0=0x98badcfe;        //C
uint32_t d0=0x10325476;        //D
uint32_t a, b, c, d, i, f, g, temp;
size_t len= 0;
float totalones=0;
vector <uint32_t> w;
#define left(x, c) (((x) << (c)) | ((x) >> (32 - (c)))); //Left shift
function

//Processing the input data,padding.
string ss;
getline(cin,ss);                    //Get input from user
vector <uint32_t> by(ss.begin(),ss.end());   //Initialise to a vector
len=by.size();
```

```
by.push_back(0x80);                     //Appending the bit '1'
for(int i=0; (by.size()*8)%512!=448;i++)      // Padding with zeros
{
    by.push_back(0x00);
}
by.push_back(len*8);                    //appending the length of the
original message

for(int i=0;(by.size()*8)%512!=0;i++)         //Appending zeros
{
    by.push_back(0x00);
}

//Breaking into blocks of 512-bit and each block is divided into 16 32-
bit word
len=by.size()/64;
int iterator=0,j=0,limit=0;
while(iterator<len){
j=limit;
limit=64*(iterator+1);
for(j=j; j<limit; j+=4)
{
        w.push_back(by[j+3]<<24|(by[j+2]<<16)|(by[j+1]<<8)|(by[j]));
//32-bit word, formed by shift operation
}

//Initial hash values
a=a0;
b=b0;
c=c0;
d=d0;

//Computing the hash values
for(int i=0;i<64;i++){
    if(i<16){
        f = (b & c) | ((~b) & d);
                g = i;
            } else if (i < 32) {
                f = (d & b) | ((~d) & c);
                g = i;
            } else if (i < 48) {
                f = b ^ c ^ d;
                g = i;
            } else {
                f = c ^ (b | (~d));
                g = i;
            }

            temp = d;
            d = c;
            c = b;
            b = b + left((a+f+keys[i]+w[g]), shifts[i]);
            a = temp;
          //cout<<w[g]<<endl;
}
iterator++;
w.clear();
        // Adding the hash values:
        a0 += a;
        b0 += b;
        c0 += c;
        d0 += d;
}
totalones=bitCount(a0)+bitCount(b0)+bitCount(c0)+bitCount(d0);
a0=reverse(a0);
b0=reverse(b0);
c0=reverse(c0);
d0=reverse(d0);
cout<<"number of ones in a0 is "<<bitset<32>(a0)<<endl;
cout<<"number of ones in b0 is "<<bitset<32>(b0)<<endl;
cout<<"number of ones in c0 is "<<bitset<32>(c0)<<endl;
cout<<"number of ones in d0 is "<<bitset<32>(d0)<<endl;
cout<<"Total Bits set "<<totalones<<endl;
cout<<"Percentage of ones is "<<((totalones*100)/128)<<endl;
cout<<"a0 is "<<hex<<a0<<endl;
cout<<"b0 is "<<hex<<b0<<endl;
cout<<"c0 is "<<hex<<c0<<endl;
cout<<"d0 is "<<hex<<d0<<endl;
cout<<"The hash value is " <<hex<<a0<<hex<<b0<<hex<<c0<<hex<<d0<<endl;
      //Final hash values
}
uint32_t reverse(uint32_t q)
{
return
(((q>>24)&0xff)|(((q>>16)&0xff)<<8)|(((q>>8)&0xff)<<16)|(((q)&0xff)<<24))
;
}
uint32_t bitCount(uint32_t n)
{
    size_t count = 0;
    while (n)
    {
      n &= (n-1) ;
      count++;
    }
    return count;
}
```