43260_main.cpp

```
#include <iostream>
#include "sha1.h"

using namespace std;

int main(int argc, char *argv[])
{
   cout << "sha1('grape'):" << sha1("grape") << endl;
   return 0;
}
```

43260_sha.cpp

```
#include "sha1.h"
#include <sstream>
#include <iomanip>
#include <fstream>

/* Help macros */
#define SHA1_ROL(value, bits) (((value) << (bits)) | (((value)&0xffffffff)
>> (32 - (bits))))
#define SHA1_BLK(i) (block[i & 15] = SHA1_ROL(block[(i + 13) & 15] ^
block[(i + 8) & 15] ^ block[(i + 2) & 15] ^ block[i & 15], 1))

/* (R0+R1), R2, R3, R4 are the different operations used in SHA1 */
#define SHA1_R0(v, w, x, y, z, i)                                    \
   z += ((w & (x ^ y)) ^ y) + block[i] + 0x5a827999 + SHA1_ROL(v, 5); \
   w = SHA1_ROL(w, 30);
#define SHA1_R1(v, w, x, y, z, i)
\
   z += ((w & (x ^ y)) ^ y) + SHA1_BLK(i) + 0x5a827999 + SHA1_ROL(v, 5); \
   w = SHA1_ROL(w, 30);
#define SHA1_R2(v, w, x, y, z, i)                               \
   z += (w ^ x ^ y) + SHA1_BLK(i) + 0x6ed9eba1 + SHA1_ROL(v, 5); \
   w = SHA1_ROL(w, 30);
#define SHA1_R3(v, w, x, y, z, i)
\
   z += (((w | x) & y) | (w & x)) + SHA1_BLK(i) + 0x8f1bbcdc + SHA1_ROL(v,
5); \
   w = SHA1_ROL(w, 30);
```

```
#define SHA1_R4(v, w, x, y, z, i)                                        \
   z += (w ^ x ^ y) + SHA1_BLK(i) + 0xca62c1d6 + SHA1_ROL(v, 5); \
   w = SHA1_ROL(w, 30);

SHA1::SHA1()
{
   reset();
}

void SHA1::update(const std::string &s)
{
   std::istringstream is(s);
   update(is);
}

void SHA1::update(std::istream &is)
{
   std::string rest_of_buffer;
   read(is, rest_of_buffer, BLOCK_BYTES - buffer.size());
   buffer += rest_of_buffer;

   while (is)
   {
       uint32 block[BLOCK_INTS];
       buffer_to_block(buffer, block);
       transform(block);
       read(is, buffer, BLOCK_BYTES);
   }
}

/*
* Add padding and return the message digest.
*/

std::string SHA1::final()
{
   /* Total number of hashed bits */
   uint64 total_bits = (transforms * BLOCK_BYTES + buffer.size()) * 8;

   /* Padding */
```

```
    buffer += 0x80;
    unsigned int orig_size = buffer.size();
    while (buffer.size() < BLOCK_BYTES)
    {
        buffer += (char)0x00;
    }

    uint32 block[BLOCK_INTS];
    buffer_to_block(buffer, block);

    if (orig_size > BLOCK_BYTES - 8)
    {
        transform(block);
        for (unsigned int i = 0; i < BLOCK_INTS - 2; i++)
        {
            block[i] = 0;
        }
    }

    /* Append total_bits, split this uint64 into two uint32 */
    block[BLOCK_INTS - 1] = total_bits;
    block[BLOCK_INTS - 2] = (total_bits >> 32);
    transform(block);

    /* Hex std::string */
    std::ostringstream result;
    for (unsigned int i = 0; i < DIGEST_INTS; i++)
    {
        result << std::hex << std::setfill('0') << std::setw(8);
        result << (digest[i] & 0xffffffff);
    }

    /* Reset for next run */
    reset();

    return result.str();
}

std::string SHA1::from_file(const std::string &filename)
{
    std::ifstream stream(filename.c_str(), std::ios::binary);
    SHA1 checksum;
    checksum.update(stream);
    return checksum.final();
}

void SHA1::reset()
{
    /* SHA1 initialization constants */
    digest[0] = 0x67452301;
    digest[1] = 0xefcdab89;
    digest[2] = 0x98badcfe;
    digest[3] = 0x10325476;
    digest[4] = 0xc3d2e1f0;

    /* Reset counters */
    transforms = 0;
    buffer = "";
}

/*
* Hash a single 512-bit block. This is the core of the algorithm.
*/

void SHA1::transform(uint32 block[BLOCK_BYTES])
{
    /* Copy digest[] to working vars */
    uint32 a = digest[0];
    uint32 b = digest[1];
    uint32 c = digest[2];
    uint32 d = digest[3];
    uint32 e = digest[4];

    /* 4 rounds of 20 operations each. Loop unrolled. */
    SHA1_R0(a, b, c, d, e, 0);
    SHA1_R0(e, a, b, c, d, 1);
    SHA1_R0(d, e, a, b, c, 2);
    SHA1_R0(c, d, e, a, b, 3);
    SHA1_R0(b, c, d, e, a, 4);
    SHA1_R0(a, b, c, d, e, 5);
    SHA1_R0(e, a, b, c, d, 6);
    SHA1_R0(d, e, a, b, c, 7);
    SHA1_R0(c, d, e, a, b, 8);
    SHA1_R0(b, c, d, e, a, 9);
    SHA1_R0(a, b, c, d, e, 10);
    SHA1_R0(e, a, b, c, d, 11);
    SHA1_R0(d, e, a, b, c, 12);
    SHA1_R0(c, d, e, a, b, 13);
    SHA1_R0(b, c, d, e, a, 14);
    SHA1_R0(a, b, c, d, e, 15);
    SHA1_R1(e, a, b, c, d, 16);
    SHA1_R1(d, e, a, b, c, 17);
    SHA1_R1(c, d, e, a, b, 18);
    SHA1_R1(b, c, d, e, a, 19);
    SHA1_R2(a, b, c, d, e, 20);
    SHA1_R2(e, a, b, c, d, 21);
    SHA1_R2(d, e, a, b, c, 22);
    SHA1_R2(c, d, e, a, b, 23);
    SHA1_R2(b, c, d, e, a, 24);
    SHA1_R2(a, b, c, d, e, 25);
    SHA1_R2(e, a, b, c, d, 26);
    SHA1_R2(d, e, a, b, c, 27);
    SHA1_R2(c, d, e, a, b, 28);
    SHA1_R2(b, c, d, e, a, 29);
    SHA1_R2(a, b, c, d, e, 30);
    SHA1_R2(e, a, b, c, d, 31);
    SHA1_R2(d, e, a, b, c, 32);
    SHA1_R2(c, d, e, a, b, 33);
    SHA1_R2(b, c, d, e, a, 34);
    SHA1_R2(a, b, c, d, e, 35);
    SHA1_R2(e, a, b, c, d, 36);
    SHA1_R2(d, e, a, b, c, 37);
    SHA1_R2(c, d, e, a, b, 38);
    SHA1_R2(b, c, d, e, a, 39);
    SHA1_R3(a, b, c, d, e, 40);
    SHA1_R3(e, a, b, c, d, 41);
    SHA1_R3(d, e, a, b, c, 42);
    SHA1_R3(c, d, e, a, b, 43);
    SHA1_R3(b, c, d, e, a, 44);
    SHA1_R3(a, b, c, d, e, 45);
    SHA1_R3(e, a, b, c, d, 46);
    SHA1_R3(d, e, a, b, c, 47);
    SHA1_R3(c, d, e, a, b, 48);
    SHA1_R3(b, c, d, e, a, 49);
    SHA1_R3(a, b, c, d, e, 50);
    SHA1_R3(e, a, b, c, d, 51);
    SHA1_R3(d, e, a, b, c, 52);
    SHA1_R3(c, d, e, a, b, 53);
    SHA1_R3(b, c, d, e, a, 54);
    SHA1_R3(a, b, c, d, e, 55);
    SHA1_R3(e, a, b, c, d, 56);
    SHA1_R3(d, e, a, b, c, 57);
    SHA1_R3(c, d, e, a, b, 58);
    SHA1_R3(b, c, d, e, a, 59);
    SHA1_R4(a, b, c, d, e, 60);
    SHA1_R4(e, a, b, c, d, 61);
    SHA1_R4(d, e, a, b, c, 62);
    SHA1_R4(c, d, e, a, b, 63);
    SHA1_R4(b, c, d, e, a, 64);
    SHA1_R4(a, b, c, d, e, 65);
    SHA1_R4(e, a, b, c, d, 66);
    SHA1_R4(d, e, a, b, c, 67);
    SHA1_R4(c, d, e, a, b, 68);
    SHA1_R4(b, c, d, e, a, 69);
    SHA1_R4(a, b, c, d, e, 70);
    SHA1_R4(e, a, b, c, d, 71);
    SHA1_R4(d, e, a, b, c, 72);
    SHA1_R4(c, d, e, a, b, 73);
    SHA1_R4(b, c, d, e, a, 74);
    SHA1_R4(a, b, c, d, e, 75);
    SHA1_R4(e, a, b, c, d, 76);
    SHA1_R4(d, e, a, b, c, 77);
    SHA1_R4(c, d, e, a, b, 78);
    SHA1_R4(b, c, d, e, a, 79);

    /* Add the working vars back into digest[] */
    digest[0] += a;
    digest[1] += b;
    digest[2] += c;
    digest[3] += d;
    digest[4] += e;

    /* Count the number of transformations */
    transforms++;
}

void SHA1::buffer_to_block(const std::string &buffer, uint32
block[BLOCK_BYTES])
{
    /* Convert the std::string (byte buffer) to a uint32 array (MSB) */
    for (unsigned int i = 0; i < BLOCK_INTS; i++)
    {
        block[i] = (buffer[4 * i + 3] & 0xff) | (buffer[4 * i + 2] & 0xff)
<< 8 | (buffer[4 * i + 1] & 0xff) << 16 | (buffer[4 * i + 0] & 0xff) <<
24;
    }
}

void SHA1::read(std::istream &is, std::string &s, int max)
{
    char sbuf[max];
    is.read(sbuf, max);
    s.assign(sbuf, is.gcount());
}

std::string sha1(const std::string &string)
{
    SHA1 checksum;
    checksum.update(string);
    return checksum.final();
}
```

43260_sha.h

```
#ifndef SHA1_HPP
#define SHA1_HPP

#include <iostream>
#include <string>

```

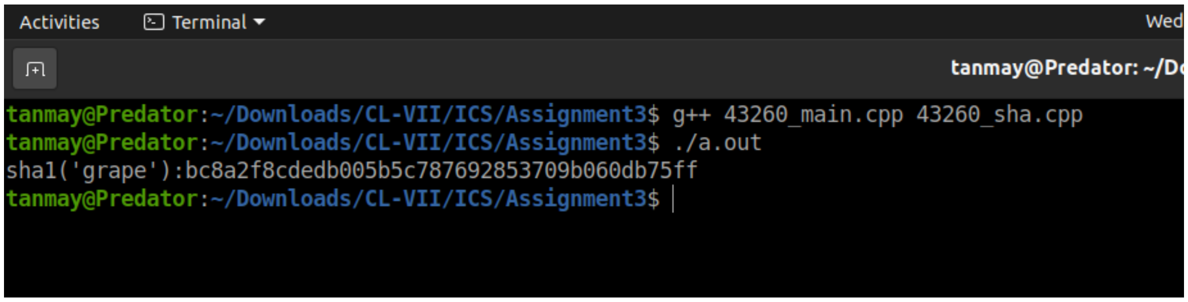
```
class SHA1
{
public:
   SHA1();
   void update(const std::string &s);
   void update(std::istream &is);
   std::string final();
   static std::string from_file(const std::string &filename);

private:
   typedef unsigned long int uint32;  /* just needs to be at least 32bit
*/
   typedef unsigned long long uint64; /* just needs to be at least 64bit
*/

   static const unsigned int DIGEST_INTS = 5; /* number of 32bit integers
per SHA1 digest */
   static const unsigned int BLOCK_INTS = 16; /* number of 32bit integers
per SHA1 block */
   static const unsigned int BLOCK_BYTES = BLOCK_INTS * 4;

   uint32 digest[DIGEST_INTS];
   std::string buffer;
   uint64 transforms;

   void reset();
   void transform(uint32 block[BLOCK_BYTES]);

   static void buffer_to_block(const std::string &buffer, uint32
block[BLOCK_BYTES]);
   static void read(std::istream &is, std::string &s, int max);
};

std::string sha1(const std::string &string);

#endif
```

43260_sha.java

```java
// Java program to calculate SHA-1 hash value

import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

class SHA {
   public static String encryptThisString(String input) {
       try { // getInstance() method is called with algorithm SHA-1
           MessageDigest md = MessageDigest.getInstance("SHA-1");
           // digest() method is called
           // to calculate message digest of the input string
           // returned as array of
           byte[] messageDigest = md.digest(input.getBytes());
           // Convert byte array into signum representation
           BigInteger no = new BigInteger(1, messageDigest);
           // Convert message digest into hex value
           String hashtext = no.toString(16);
           // Add preceding 0s to make it 32 bit
           while (hashtext.length() < 32) {
               hashtext = "0" + hashtext;
           }
           // return the HashText
           return hashtext;
       }
       // For specifying wrong message digest algorithms
       catch (NoSuchAlgorithmException e) {
           throw new RuntimeException(e);
       }
   }

   // Driver code
   public static void main(String args[]) throws NoSuchAlgorithmException
{
       System.out.println("HashCode Generated by SHA-1 for: ");
       String s1 = "JAVA Program";
       System.out.println(s1 + " : " + encryptThisString(s1));
       String s2 = "Hello World";
       System.out.println(s2 + " : " + encryptThisString(s2));
```

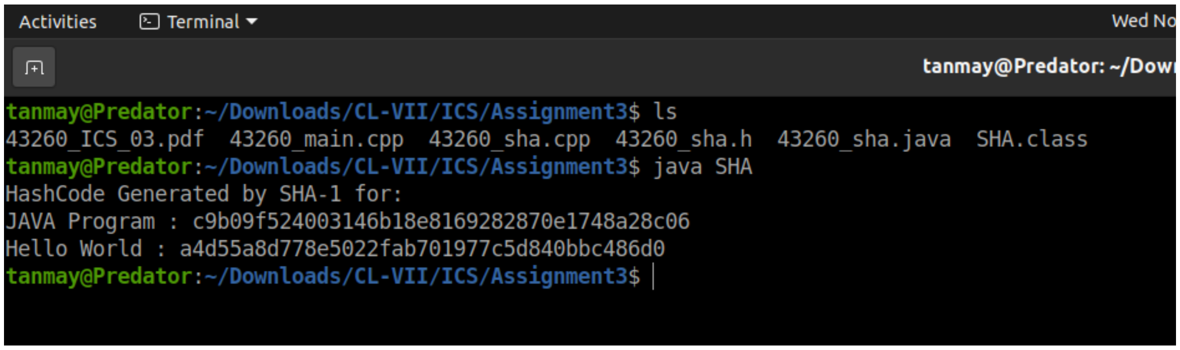
```
    }
}
```

CPP Output:



```
tanmay@Predator:~/Downloads/CL-VII/ICS/Assignment3$ g++ 43260_main.cpp 43260_sha.cpp
tanmay@Predator:~/Downloads/CL-VII/ICS/Assignment3$ ./a.out
sha1('grape'):bc8a2f8cdedb005b5c787692853709b060db75ff
tanmay@Predator:~/Downloads/CL-VII/ICS/Assignment3$
```

Java Output:



```
tanmay@Predator:~/Downloads/CL-VII/ICS/Assignment3$ ls
43260_ICS_03.pdf  43260_main.cpp  43260_sha.cpp  43260_sha.h  43260_sha.java  SHA.class
tanmay@Predator:~/Downloads/CL-VII/ICS/Assignment3$ java SHA
HashCode Generated by SHA-1 for:
JAVA Program : c9b09f524003146b18e8169282870e1748a28c06
Hello World : a4d55a8d778e5022fab701977c5d840bbc486d0
tanmay@Predator:~/Downloads/CL-VII/ICS/Assignment3$
```