Submitted By,
Shivaprasad B J
201948023

AC LAB-6

1) Write C Program to implement SHA-1 hash technique.

Code:

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

#define rotateleft(x,n) ((x<<n) | (x>>(32-n)))
#define rotateright(x,n) ((x>>n) | (x<<(32-n)))

void SHA1(unsigned char * str1)
{
    unsigned long int h0,h1,h2,h3,h4,a,b,c,d,e,f,k,temp;
    int i,j;

    h0 = 0x67452301;
    h1 = 0xEFCDAB89;
    h2 = 0x98BADCFE;
    h3 = 0x10325476;
    h4 = 0xC3D2E1F0;

    unsigned char * str;
    str = (unsigned char *)malloc(strlen((const char *)str1)+100);
    strcpy((char *)str,(const char *)str1);

    int current_length = strlen((const char *)str);
    int original_length = current_length;
    str[current_length] = 0x80;
    str[current_length + 1] = '\0';

    char ic = str[current_length];
    current_length++;

    int ib = current_length % 64;
    if(ib<56)
        ib = 56-ib;
    else
        ib = 120 - ib;

    for(i=0;i < ib;i++)
    {
        str[current_length]=0x00;
```

Submitted By,
Shivaprasad B J
201948023

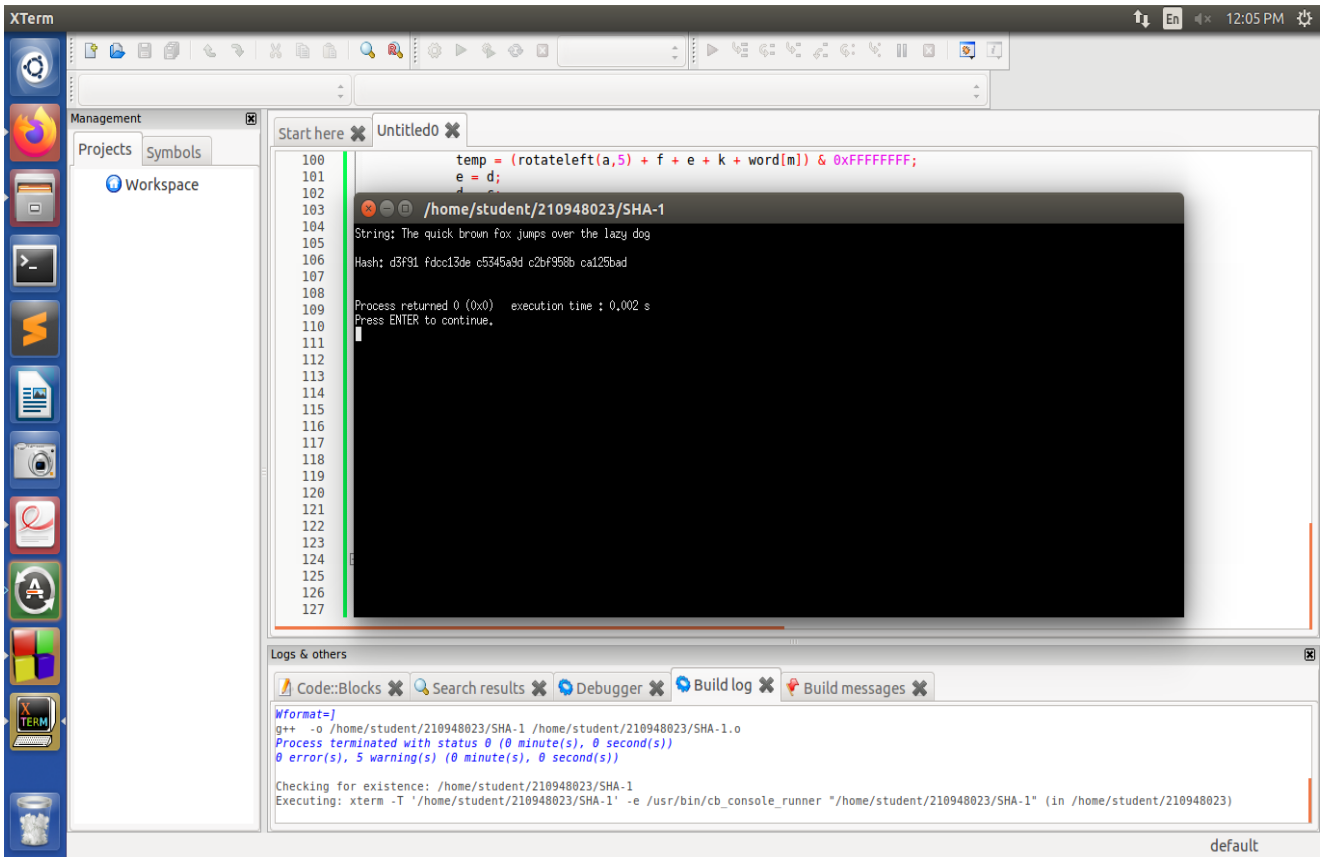
```
        current_length++;
    }
    str[current_length + 1]='\0';

    for(i=0;i<6;i++)
    {
        str[current_length]=0x0;
        current_length++;
    }
    str[current_length] = (original_length * 8) / 0x100 ;
    current_length++;
    str[current_length] = (original_length * 8) % 0x100;
    current_length++;
    str[current_length+i]='\0';

    int number_of_chunks = current_length/64;
    unsigned long int word[80];
    for(i=0;i<number_of_chunks;i++)
    {
        for(j=0;j<16;j++)
        {
            word[j] = str[i*64 + j*4 + 0] * 0x1000000 + str[i*64 + j*4 + 1] *
0x10000 + str[i*64 + j*4 + 2] * 0x100 + str[i*64 + j*4 + 3];
        }
        for(j=16;j<80;j++)
        {
            word[j] = rotateleft((word[j-3] ^ word[j-8] ^ word[j-14] ^ word[j-
16]),1);
        }

        a = h0;
        b = h1;
        c = h2;
        d = h3;
        e = h4;

        for(int m=0;m<80;m++)
        {
            if(m<=19)
            {
                f = (b & c) | ((~b) & d);
                k = 0x5A827999;
            }
            else if(m<=39)
            {
                f = b ^ c ^ d;
                k = 0x6ED9EBA1;
            }
```

Submitted By,
Shivaprasad B J
201948023

```
            else if(m<=59)
            {
                f = (b & c) | (b & d) | (c & d);
                k = 0x8F1BBCDC;
            }
            else
            {
                f = b ^ c ^ d;
                k = 0xCA62C1D6;
            }

            temp = (rotateleft(a,5) + f + e + k + word[m]) & 0xFFFFFFFF;
            e = d;
            d = c;
            c = rotateleft(b,30);
            b = a;
            a = temp;

        }

        h0 = h0 + a;
        h1 = h1 + b;
        h2 = h2 + c;
        h3 = h3 + d;
        h4 = h4 + e;

    }
    printf("String: The quick brown fox jumps over the lazy dog");
    printf("\n\n");

    printf("Hash: %x %x %x %x %x",h0, h1, h2, h3, h4);
    printf("\n\n");
}

void main()
{
    SHA1((unsigned char *)"The quick brown fox jumps over the lazy dog");
}
```

Submitted By,
Shivaprasad B J
201948023

Output:



Submitted By,
Shivaprasad B J
201948023

2) Write C program to implement Digital signature using RSA.

Code:

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

typedef struct {
 int d;
 int x;
 int y;
} EE;

EE extended_euclid(int a, int b) {
 EE ee1, ee2, ee3;
 if (b == 0) {
  ee1.d = a;
  ee1.x = 1;
  ee1.y = 0;
  return ee1;
 } else {
  ee2 = extended_euclid(b, a % b);
  ee3.d = ee2.d;
  ee3.x = ee2.y;
  ee3.y = ee2.x - floor(a / b) * ee2.y;
  return ee3;
 }
}

// Copied from
```

Submitted By,
Shivaprasad B J
201948023

```
// https://stackoverflow.com/questions/11720656/modulo-operation-with-negative-numbers
int modulo(int x, int N){
    return (x % N + N) % N;
}

void decimal_to_binary(int op1, int aOp[]){
    int result, i = 0;
    do{
        result = op1 % 2;
        op1 /= 2;
        aOp[i] = result;
        i++;
    }while(op1 > 0);
}

int modular_exponentiation(int a, int b, int n){
 int *bb;
 int count = 0, c = 0, d = 1, i;

 // find out the size of binary b
 count = (int) (log(b)/log(2)) + 1;

 bb = (int *) malloc(sizeof(int*) * count);
 decimal_to_binary(b, bb);

 for (i = count - 1; i >= 0; i--) {
  c = 2 * c;
  d = (d*d) % n;
```

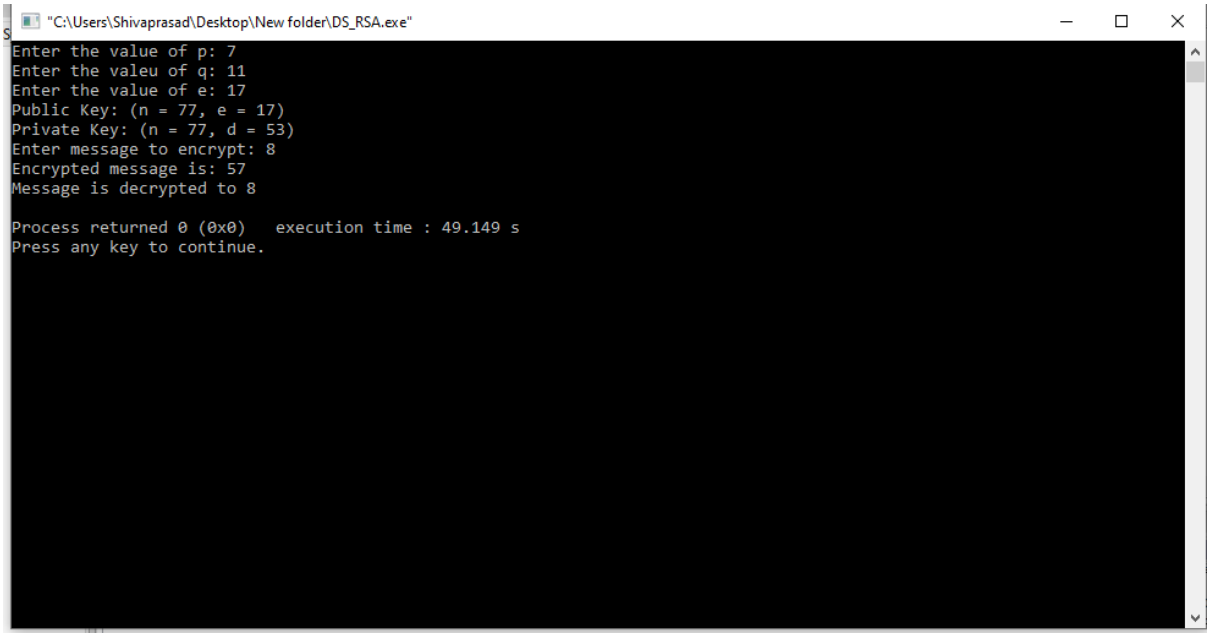
Submitted By,
Shivaprasad B J
201948023

```
 if (bb[i] == 1) {
  c = c + 1;
  d = (d*a) % n;
 }
}
return d;
}

int get_d(int e, int phi){
 EE ee;
 ee = extended_euclid(e, phi);
 return modulo(ee.x, phi);
}

int main(int argc, char* argv[]) {
 int p, q, phi, n, e, d, m, c;
 printf("Enter the value of p: ");
 scanf("%d", &p);
 printf("Enter the valeu of q: ");
 scanf("%d", &q);
 n = p*q;
 phi = (p - 1) * (q - 1);
 printf("Enter the value of e: ");
 scanf("%d", &e);
 d = get_d(e, phi);
 printf("Public Key: (n = %d, e = %d)\n", n, e);
 printf("Private Key: (n = %d, d = %d)\n", n, d);
 printf("Enter message to encrypt: ");
 scanf("%d", &m);
```

Submitted By,
Shivaprasad B J
201948023

```
 c = modular_exponentiation(m, e, n);
 printf("Encrypted message is: %d\n", c);
 m = modular_exponentiation(c, d, n);
 printf("Message is decrypted to %d\n", m);
 return 0;
}
```

OUTPUT:



Submitted By,
Shivaprasad B J
201948023

3) C program to implement Digital Signature using El gamal.

CODE:

```
import random
from math import pow
a = random.randint(2, 10)

def gcd(a, b):
    if a < b:
        return gcd(b, a)
    elif a % b == 0:
        return b;
    else:
        return gcd(b, a % b)
def gen_key(q):
    key = random.randint(pow(10, 20), q)
    while gcd(q, key) != 1:
        key = random.randint(pow(10, 20), q)
    return key

def power(a, b, c):
    x = 1
    y = a
    while b > 0:
        if b % 2 != 0:
            x = (x * y) % c;
        y = (y * y) % c
        b = int(b / 2)
    return x % c
```

Submitted By,
Shivaprasad B J
201948023

```
def encrypt(msg, q, h, g):
    en_msg = []
    k = gen_key(q)# Private key for sender
    s = power(h, k, q)
    p = power(g, k, q)

    for i in range(0, len(msg)):
        en_msg.append(msg[i])
    print("g^k used : ", p)
    print("g^ak used : ", s)
    for i in range(0, len(en_msg)):
        en_msg[i] = s * ord(en_msg[i])
    return en_msg, p

def decrypt(en_msg, p, key, q):
    dr_msg = []
    h = power(p, key, q)
    for i in range(0, len(en_msg)):
        dr_msg.append(chr(int(en_msg[i]/h)))
    return dr_msg

# Driver code
def main():
    msg = 'encryption'
    print("Original Message :", msg)
    q = random.randint(pow(10, 20), pow(10, 50))
    g = random.randint(2, q)
    key = gen_key(q)# Private key for receiver
    h = power(g, key, q)
    print("g used : ", g)
    print("g^a used : ", h)
```

Submitted By,
Shivaprasad B J
201948023

```
        en_msg, p = encrypt(msg, q, h, g)
        dr_msg = decrypt(en_msg, p, key, q)
        dmsg = ''.join(dr_msg)
        print("Decrypted Message :", dmsg);

if __name__ == '__main__':
        main()
```