Government Engineering College
Sec-28 Gandhinagar

Sem: - VII

Subject: - Information Security

Subject Code: - 3170720

Government Engineering College

Sec-28 Gandhinagar

Certificate

This is to certify that

Mr./Ms. ………..**Italiya Nirajkumar Vijaybhai**……………….. *Of class*

……CE…… Division …A……, Enrollment No. …190130107041.. Has

Satisfactorily completed his/her term work in… **Information Security**

………. Subject for the term ending in ……………2022.

Date: -

Vision and Mission

Institute (GECG):

Vision:	To be a premier engineering institution, imparting quality education for innovative solutions relevant to society and environment.
Mission:	● To develop human potential to its fullest extent so that intellectual and innovative engineers can emerge in a wide range of professions. ● To advance knowledge and educate students in engineering and other areas of scholarship that will best serve the nation and the world in future. ● To produce quality engineers, entrepreneurs and leaders to meet the present and future needs of society as well as the environment.

Department (CE):

Vision:	To achieve excellence for providing value based education in Computer Engineering through innovation, teamwork and ethical practices.
Mission:	● To produce computer science and engineering graduates according to the needs of industry, government, society and scientific community. ● To develop partnership with industries, government agencies and R and D Organizations ● To motivate students/graduates to be entrepreneurs. ● To motivate students to participate in reputed conferences, workshops, symposiums, seminars and related technical activities

Course Outcomes:

Sr. No.	CO statement	Marks % weightage
CO-1	Explore the basic principles of the symmetric cryptography and techniques with their strengths and weaknesses from perspective of cryptanalysis	10
CO-2	Implement and analyze various symmetric key cryptography algorithms and their application in different context.	25
CO-3	Compare public key cryptography with private key cryptography and Implement various asymmetric key cryptography algorithms.	25
CO-4	Explore the concept of hashing and implement various hashing algorithms for message integrity.	20
CO-5	Explore and use the techniques and standards of digital signature, key management and authentication.	20

Index

Practical-1: Implement Encryption-Decryption for Caesar cipher and brute force attack on Caesar cipher.

Code:

```
#include<stdio.h>

#include<string.h>

#include<time.h>

int main()

{

   char message[100], ch;

   time_t t; // not a primitive datatype

   time(&t);

   int i, key;


   printf("190130107041\n");

   printf("Niraj Italiya\n");

   printf("Prac 1-A\n");

   printf("Implement Encryption-Decryption for Caesar cipher\n");

   printf("%s\n\n", ctime(&t));


   printf("Enter a message to encrypt: ");

   gets(message);
```

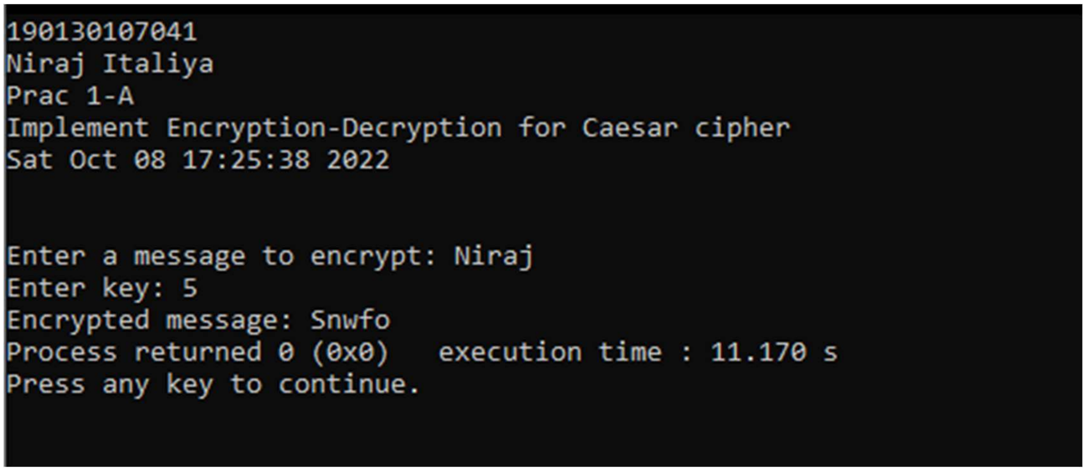
```
printf("Enter key: ");

scanf("%d", &key);


for(i = 0; message[i] != '\0'; ++i)

{


    ch = message[i];

    if(ch >= 'a' && ch <= 'z')

    {

        ch = ch + key;

        if(ch > 'z')

        {

            ch = ch - 'z' + 'a' - 1;

        }

        message[i] = ch;

    }

    else if(ch >= 'A' && ch <= 'Z')

    {

        ch = ch + key;


        if(ch > 'Z')

        {

            ch = ch - 'Z' + 'A' - 1;

        }

    message[i] = ch;
```

```
        }
    }
    printf("Encrypted message: %s", message);
    return 0;
}
```

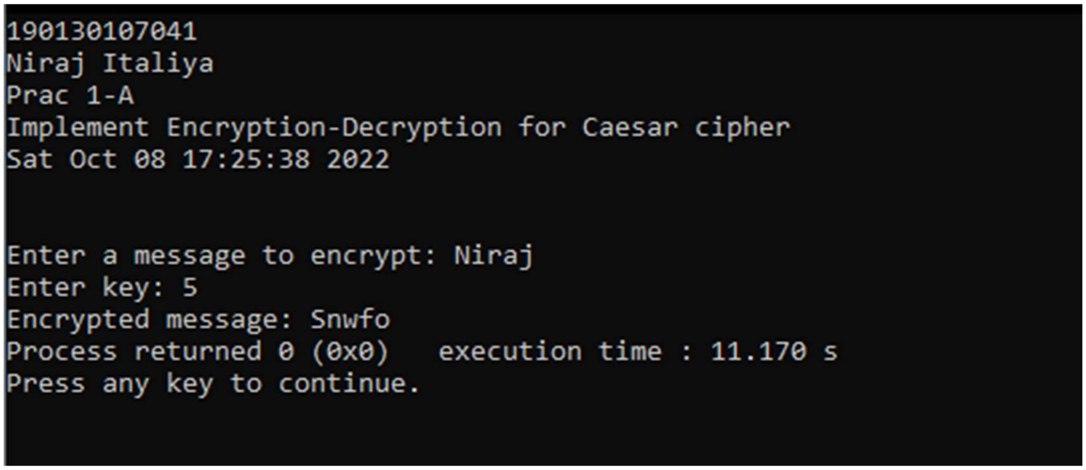
Decryption :

```
#include<stdio.h>
int main()
{
    char message[100], ch;
    int i, key;

    printf("190130107041\n");
    printf("Niraj Italiya \n");
    printf("Prac 1-A\n");
    printf("Implement Encryption-Decryption for Caesar cipher\n\n");

    printf("Enter a message to decrypt: ");
    gets(message);

    printf("Enter key: ");
    scanf("%d", &key);

    for(i = 0; message[i] != '\0'; ++i)
    {
        ch = message[i];
        if(ch >= 'a' && ch <= 'z'){

            ch = ch - key;
            if(ch < 'a'){

                ch = ch + 'z' - 'a' + 1;
            }
            message[i] = ch;
        }
        else if(ch >= 'A' && ch <= 'Z')
        {
            ch = ch - key;
            if(ch < 'A'){
                ch = ch + 'Z' - 'A' + 1;
            }
            message[i] = ch;
        }
    }
    printf("Decrypted message: %s", message);
    return 0;
}
```

Program Execution Snapshot :



```
190130107041
Niraj Italiya
Prac 1-A
Implement Encryption-Decryption for Caesar cipher
Sat Oct 08 17:25:38 2022


Enter a message to encrypt: Niraj
Enter key: 5
Encrypted message: Snwfo
Process returned 0 (0x0)   execution time : 11.170 s
Press any key to continue.
```



```
190130107041
Niraj Italiya
Prac 1-A
Implement Encryption-Decryption for Caesar cipher
Sat Oct 08 17:25:38 2022


Enter a message to encrypt: Niraj
Enter key: 5
Encrypted message: Snwfo
Process returned 0 (0x0)   execution time : 11.170 s
Press any key to continue.
```

<u>Practical-1B</u>: Implement encryption and decryption of brute force attack on Caesar cipher.

Code:

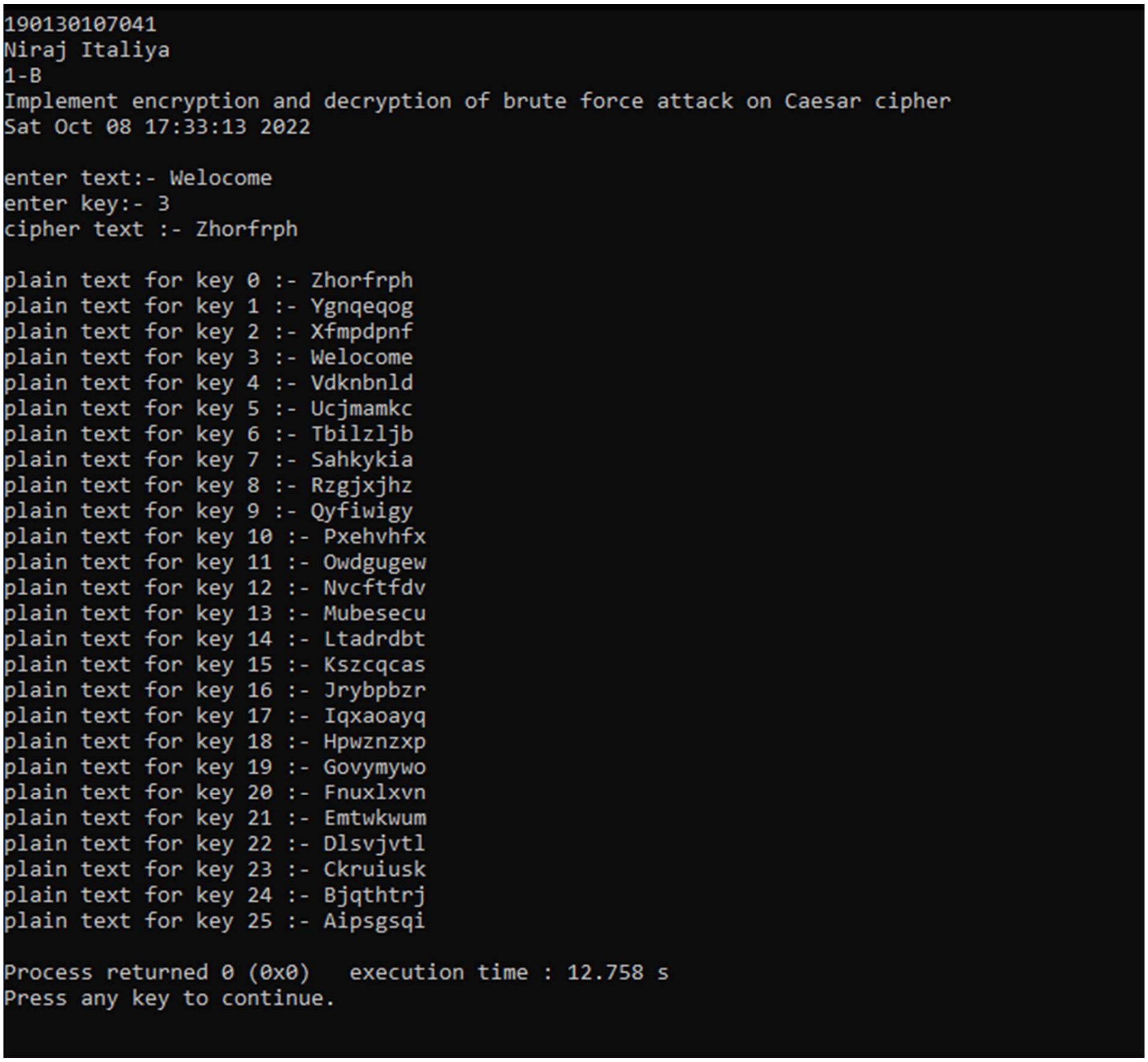
```
#include<iostream>

#include<time.h>



using namespace std;

//function to encrypt the plain text

string encrypt(string x,int n)
```

```
{
   string cipher="";


 /* only caps and small caps alphabet would be considered for encryption other symbols would
remain as it is.
*/


   for(int i=0;i<x.length();i++)
   {
      if(isupper(x[i]))
         cipher += (x[i] + n - 65)%26 + 65;
         /* here x[i] would be ASCII value of corresponding alphabet */
      else if(islower(x[i]))
         cipher += (x[i] + n - 97)%26 + 97;
      else
         cipher += x[i];
         /* other symbols other than alphabets would remain as it is. */
   }
   return cipher;
 }
//function to decrypt the cipher text using brute force attack
void decrypt(string x)
{
   string text;
   for(int n=0;n<26;n++)
```

```
    {
        text = "";
        for(int i=0;i<x.length();i++)
        {
            if(isupper(x[i]))
            {
                if((x[i] - n - 65)<0)
                    text += 91 + (x[i] - n - 65);
                else
                    text += (x[i] - n - 65)%26 + 65;
            }
            else if(islower(x[i]))
            {
                if((x[i] - n - 97) < 0)
                    text += 123 + (x[i] - n - 97);
                else
                    text += (x[i] - n - 97)%26 + 97;
            }
            else
            text += x[i];
        }
        cout << "plain text for key " << n << " :- " << text << endl;
    }
 }
int main()
```

```
{
    int key;
    string text;
    time_t t; // not a primitive datatype
    time(&t);

    cout << "190130107041\n";
    cout << "Niraj Italiya\n";
    cout << "1-B\n";
    cout << "Implement encryption and decryption of brute force attack on Caesar cipher\n";
    cout << ("%s\n\n", ctime(&t));

    cout << "\nenter text:- ";
    getline(cin,text);

    cout << "enter key:- ";
    cin >> key;

    string cipher = encrypt(text,key);
    cout << "cipher text :- " << cipher << endl << endl;
    decrypt(cipher);
}
```

Program Execution Snapshot :



```
190130107041
Niraj Italiya
1-B
Implement encryption and decryption of brute force attack on Caesar cipher
Sat Oct 08 17:33:13 2022

enter text:- Welocome
enter key:- 3
cipher text :- Zhorfrph

plain text for key 0 :- Zhorfrph
plain text for key 1 :- Ygnqeqog
plain text for key 2 :- Xfmpdpnf
plain text for key 3 :- Welocome
plain text for key 4 :- Vdknbnld
plain text for key 5 :- Ucjmamkc
plain text for key 6 :- Tbilzljb
plain text for key 7 :- Sahkykia
plain text for key 8 :- Rzgjxjhz
plain text for key 9 :- Qyfiwigy
plain text for key 10 :- Pxehvhfx
plain text for key 11 :- Owdgugew
plain text for key 12 :- Nvcftfdv
plain text for key 13 :- Mubesecu
plain text for key 14 :- Ltadrdbt
plain text for key 15 :- Kszcqcas
plain text for key 16 :- Jrybpbzr
plain text for key 17 :- Iqxaoayq
plain text for key 18 :- Hpwznzxp
plain text for key 19 :- Govymywo
plain text for key 20 :- Fnuxlxvn
plain text for key 21 :- Emtwkwum
plain text for key 22 :- Dlsvjvtl
plain text for key 23 :- Ckruiusk
plain text for key 24 :- Bjqthtrj
plain text for key 25 :- Aipsgsqi

Process returned 0 (0x0)   execution time : 12.758 s
Press any key to continue.
```

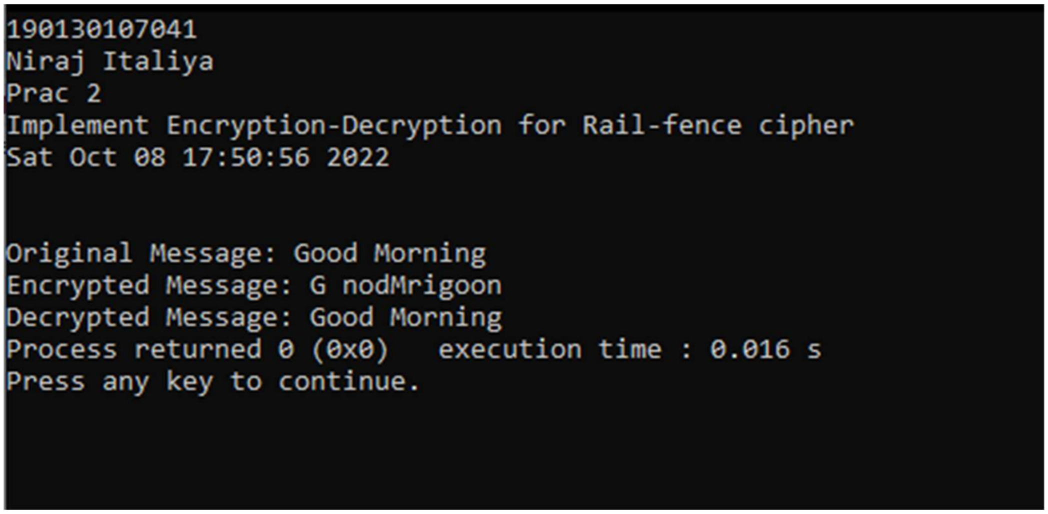
Practical-2: Implement Rail-fence cipher encryption-decryption.

Code :

```
#include<stdio.h>

#include<string.h>

#include<time.h>

void encryptMsg(char msg[], int key)

{

   int msgLen = strlen(msg), i, j, k = -1, row = 0, col = 0;

   char railMatrix[key][msgLen];

   for(i = 0; i < key; ++i)

      for(j = 0; j < msgLen; ++j)

         railMatrix[i][j] = '\n';

   for(i = 0; i < msgLen; ++i)

   {

      railMatrix[row][col++] = msg[i];

      if(row == 0 || row == key-1)

         k= k * (-1);

      row = row + k;

   }


   printf("\nEncrypted Message: ");


   for(i = 0; i < key; ++i)
```

```
        for(j = 0; j < msgLen; ++j)
            if(railMatrix[i][j] != '\n')
                printf("%c", railMatrix[i][j]);
}

void decryptMsg(char enMsg[], int key)
{
    int msgLen = strlen(enMsg), i, j, k = -1, row = 0, col = 0, m = 0;
    char railMatrix[key][msgLen];
    for(i = 0; i < key; ++i)
        for(j = 0; j < msgLen; ++j)
            railMatrix[i][j] = '\n';

    for(i = 0; i < msgLen; ++i)
    {
        railMatrix[row][col++] = '*';
        if(row == 0 || row == key-1)
            k= k * (-1);
        row = row + k;
    }

    for(i = 0; i < key; ++i)
        for(j = 0; j < msgLen; ++j)
            if(railMatrix[i][j] == '*')
                railMatrix[i][j] = enMsg[m++];
    row = col = 0;
    k = -1;

    printf("\nDecrypted Message: ");

    for(i = 0; i < msgLen; ++i){
        printf("%c", railMatrix[row][col++]);
        if(row == 0 || row == key-1)
            k= k * (-1);
        row = row + k;
    }
}

// Driver code
int main()
{
    char msg[] = "Good Morning";
    char enMsg[] = "G nodMrigoon";
    int key = 3;
    time_t t; // not a primitive datatype
    time(&t);

    printf("190130107041\n");
    printf("Niraj Italiya\n");
    printf("Prac 2\n");
    printf("Implement Encryption-Decryption for Rail-fence cipher\n");
    printf("%s\n\n", ctime(&t));


    printf("Original Message: %s", msg);


    encryptMsg(msg, key);
    decryptMsg(enMsg, key);
    return 0;
}
```

Program Execution Snapshot :



```
190130107041
Niraj Italiya
Prac 2
Implement Encryption-Decryption for Rail-fence cipher
Sat Oct 08 17:50:56 2022


Original Message: Good Morning
Encrypted Message: G nodMrigoon
Decrypted Message: Good Morning
Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.
```

<u>Practical-3:</u> Implement Playfair cipher encryption-decryption.

Code:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

#define SIZE 30


// Function to convert the string to lowercase

void toLowerCase(char plain[], int ps)

{

    int i;

    for (i = 0; i < ps; i++)

    {

        if (plain[i] > 64 && plain[i] < 91)
```

```
            plain[i] += 32;
    }
}
// Function to remove all spaces in a string
int removeSpaces(char* plain, int ps){

    int i, count = 0;
    for (i = 0; i < ps; i++)
        if (plain[i] != ' ')
            plain[count++] = plain[i];

    plain[count] = '\0';
    return count;

}
// Function to generate the 5x5 key square
void generateKeyTable(char key[], int ks, char keyT[5][5]){

    int i, j, k, flag = 0, *dicty;

    // a 26 character hashmap
    // to store count of the alphabet
    dicty = (int*)calloc(26, sizeof(int));

    for (i = 0; i < ks; i++)
```

```
{
    if (key[i] != 'j')
    dicty[key[i] - 97] = 2;
}
dicty['j' - 97] = 1;

i = 0;
j = 0;

for (k = 0; k < ks; k++)
{
    if (dicty[key[k] - 97] == 2)
    {

        dicty[key[k] - 97] -= 1;
        keyT[i][j] = key[k];
        j++;

        if (j == 5) {

            i++;
            j = 0;
        }
    }
}
```

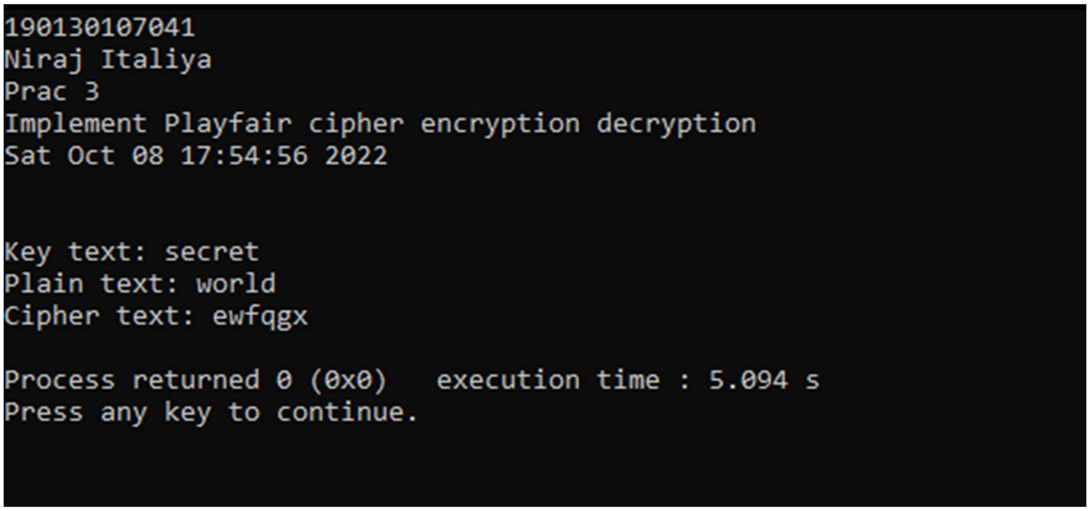
```
    for (k = 0; k < 26; k++)
    {
        if (dicty[k] == 0) {

            keyT[i][j] = (char)(k + 97);
            j++;

            if (j == 5) {
                i++;
                j = 0;
            }
        }
    }
}

// Function to search for the characters of a digraph
// in the key square and return their position
void search(char keyT[5][5], char a, char b, int arr[])
{
    int i, j;

    if (a == 'j')
        a = 'i';
    else if (b == 'j')
```

```
        b = 'i';

    for (i = 0; i < 5; i++) {

        for (j = 0; j < 5; j++)
        {
            if (keyT[i][j] == a) {
                arr[0] = i;
                arr[1] = j;
            }

            else if (keyT[i][j] == b)
            {
                arr[2] = i;
                arr[3] = j;
            }
        }
    }
}

// Function to find the modulus with 5
int mod5(int a)
{
    return (a % 5);
}
```

```
// Function to make the plain text length to be even
int prepare(char str[], int ptrs)
{
   if (ptrs % 2 != 0) {
      str[ptrs++] = 'z';
      str[ptrs] = '\0';
   }
   return ptrs;
}
// Function for performing the encryption
void encrypt(char str[], char keyT[5][5], int ps)
{
   int i, a[4];
   for (i = 0; i < ps; i += 2)
   {
      search(keyT, str[i], str[i + 1], a);

      if (a[0] == a[2])
      {
         str[i] = keyT[a[0]][mod5(a[1] + 1)];
         str[i + 1] = keyT[a[0]][mod5(a[3] + 1)];
      }
      else if (a[1] == a[3]) {

         str[i] = keyT[mod5(a[0] + 1)][a[1]];
```

```
            str[i + 1] = keyT[mod5(a[2] + 1)][a[1]];
        }
        else {

            str[i] = keyT[a[0]][a[3]];
            str[i + 1] = keyT[a[2]][a[1]];
        }
    }
}
// Function to encrypt using Playfair Cipher
void encryptByPlayfairCipher(char str[], char key[])
{
    char ps, ks, keyT[5][5];

    // Key
    ks = strlen(key);
    ks = removeSpaces(key, ks);

    toLowerCase(key, ks);
    // Plaintext
    ps = strlen(str);
    toLowerCase(str, ps);
    ps = removeSpaces(str, ps);
    ps = prepare(str, ps);
    generateKeyTable(key, ks, keyT);
    encrypt(str, keyT, ps);
}

// Driver code
int main()
{
    char str[SIZE], key[SIZE];

    // Key to be encrypted
    strcpy(key, "secret");
    time_t t; // not a primitive datatype
    time(&t);

    printf("190130107041\n");
    printf("Niraj Italiya\n");
    printf("Prac 3\n");
    printf("Implement Playfair cipher encryption decryption\n");
    printf("%s\n\n", ctime(&t));

    printf("Key text: %s\n", key);
    // Plaintext to be encrypted
    strcpy(str, "world");
    printf("Plain text: %s\n", str);
    // encrypt using Playfair Cipher

    encryptByPlayfairCipher(str, key);

    printf("Cipher text: %s\n", str);

    return 0;

}
```

Program Execution Snapshot:



```
190130107041
Niraj Italiya
Prac 3
Implement Playfair cipher encryption decryption
Sat Oct 08 17:54:56 2022


Key text: secret
Plain text: world
Cipher text: ewfqgx

Process returned 0 (0x0)   execution time : 5.094 s
Press any key to continue.
```

Practical-4: Implement key generation of DES..

OR

Practical-5: Implement DES (Data Encryption Standard).

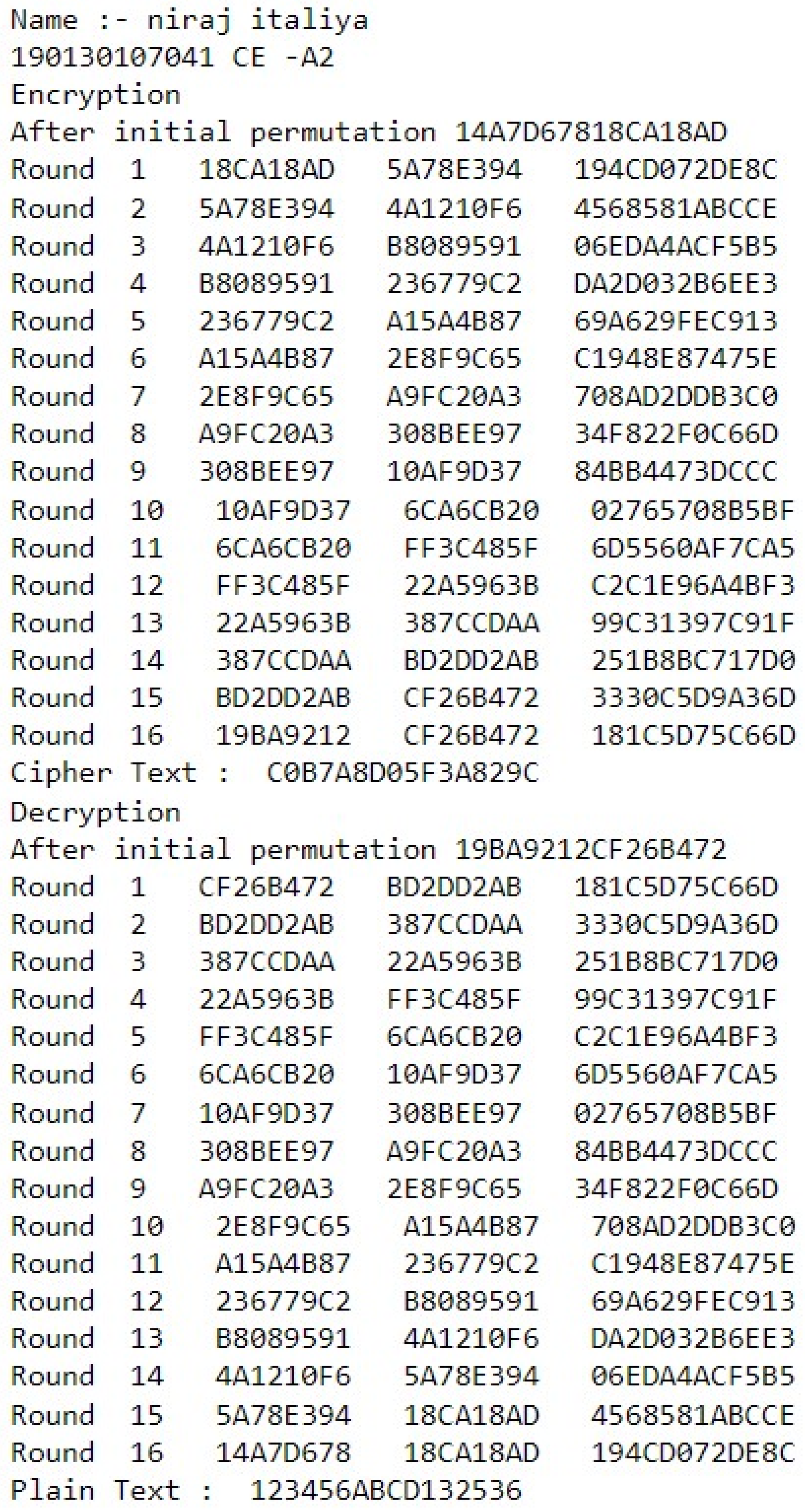
Code:

```
def hex2bin(s):
    mp = {'0': "0000",
          '1': "0001",
          '2': "0010",
          '3': "0011",
          '4': "0100",
          '5': "0101",
          '6': "0110",
          '7': "0111",
          '8': "1000",
          '9': "1001",
          'A': "1010",
          'B': "1011",
          'C': "1100",
          'D': "1101",
          'E': "1110",
          'F': "1111"}
    bin = ""
    for i in range(len(s)):
        bin = bin + mp[s[i]]
    return bin

# Binary to hexadecimal conversion



def bin2hex(s):
    mp = {"0000": '0',
          "0001": '1',
          "0010": '2',
          "0011": '3',
          "0100": '4',
          "0101": '5',
          "0110": '6',
          "0111": '7',
          "1000": '8',
          "1001": '9',
          "1010": 'A',
          "1011": 'B',
          "1100": 'C',
          "1101": 'D',
          "1110": 'E',
          "1111": 'F'}
    hex = ""
    for i in range(0, len(s), 4):
        ch = ""
        ch = ch + s[i]
        ch = ch + s[i + 1]
        ch = ch + s[i + 2]
        ch = ch + s[i + 3]
        hex = hex + mp[ch]

    return hex
def bin2dec(binary):


    binary1 = binary
    decimal, i, n = 0, 0, 0
    while(binary != 0):
        dec = binary % 10
        decimal = decimal + dec * pow(2, i)
        binary = binary//10
        i += 1
    return decimal
def dec2bin(num):
    res = bin(num).replace("0b", "")
    if(len(res) % 4 != 0):
        div = len(res) / 4
        div = int(div)
        counter = (4 * (div + 1)) - len(res)
        for i in range(0, counter):
            res = '0' + res
    return res
def permute(k, arr, n):
    permutation = ""
    for i in range(0, n):
        permutation = permutation + k[arr[i] - 1]
    return permutation
def shift_left(k, nth_shifts):
    s = ""
    for i in range(nth_shifts):
        for j in range(1, len(k)):
            s = s + k[j]
        s = s + k[0]
        k = s
        s = ""
    return k
def xor(a, b):
    ans = ""
    for i in range(len(a)):
        if a[i] == b[i]:
            ans = ans + "0"
        else:
            ans = ans + "1"
    return ans
initial_perm = [58, 50, 42, 34, 26, 18, 10, 2,
                60, 52, 44, 36, 28, 20, 12, 4,
                62, 54, 46, 38, 30, 22, 14, 6,
                64, 56, 48, 40, 32, 24, 16, 8,
                57, 49, 41, 33, 25, 17, 9, 1,
                59, 51, 43, 35, 27, 19, 11, 3,
                61, 53, 45, 37, 29, 21, 13, 5,
                63, 55, 47, 39, 31, 23, 15, 7]

# Expansion D-box Table
exp_d = [32, 1, 2, 3, 4, 5, 4, 5,
         6, 7, 8, 9, 8, 9, 10, 11,
         12, 13, 12, 13, 14, 15, 16, 17,
         16, 17, 18, 19, 20, 21, 20, 21,
         22, 23, 24, 25, 24, 25, 26, 27,
         28, 29, 28, 29, 30, 31, 32, 1]
```

```
# Straight Permutation Table

per = [16,  7, 20, 21,

       29, 12, 28, 17,

       1, 15, 23, 26,

       5, 18, 31, 10,

       2,  8, 24, 14,

       32, 27,  3,  9,

       19, 13, 30,  6,

       22, 11,  4, 25]


# S-box Table

sbox = [[[14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7],

         [0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8],

         [4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0],

         [15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13]],


        [[15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10],

         [3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5],

         [0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15],

         [13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9]],


        [[10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8],

         [13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1],
         [13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7],
         [1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12]],

        [[7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15],
         [13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9],
         [10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4],
         [3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14]],

        [[2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9],
         [14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6],
         [4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14],
         [11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3]],

        [[12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11],
         [10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8],
         [9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6],
         [4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13]],

        [[4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1],
         [13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6],
         [1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2],
         [6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12]],
        [[13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7],
         [1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2],
         [7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8],
         [2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11]]]

# Final Permutation Table
final_perm = [40, 8, 48, 16, 56, 24, 64, 32,
              39, 7, 47, 15, 55, 23, 63, 31,
              38, 6, 46, 14, 54, 22, 62, 30,
              37, 5, 45, 13, 53, 21, 61, 29,
              36, 4, 44, 12, 52, 20, 60, 28,
              35, 3, 43, 11, 51, 19, 59, 27,
              34, 2, 42, 10, 50, 18, 58, 26,
              33, 1, 41, 9, 49, 17, 57, 25]
def encrypt(pt, rkb, rk):
    pt = hex2bin(pt)
    # Initial Permutation
    pt = permute(pt, initial_perm, 64)
    print("After initial permutation", bin2hex(pt))
    # Splitting
    left = pt[0:32]
    right = pt[32:64]
    for i in range(0, 16):
        #  Expansion D-box: Expanding the 32 bits data into 48 bits
        right_expanded = permute(right, exp_d, 48)
        # XOR RoundKey[i] and right_expanded
        xor_x = xor(right_expanded, rkb[i])

        # S-boxex: substituting the value from s-box table by calculating row and column
        sbox_str = ""
        for j in range(0, 8):
            row = bin2dec(int(xor_x[j * 6] + xor_x[j * 6 + 5]))
            col = bin2dec(
                int(xor_x[j * 6 + 1] + xor_x[j * 6 + 2] + xor_x[j * 6 + 3] + xor_x[j * 6 + 4]))
            val = sbox[j][row][col]
            sbox_str = sbox_str + dec2bin(val)

        # Straight D-box: After substituting rearranging the bits
        sbox_str = permute(sbox_str, per, 32)

        # XOR left and sbox_str
        result = xor(left, sbox_str)
        left = result
        if(i != 15):
            left, right = right, left
        print("Round ", i + 1, " ", bin2hex(left),
              " ", bin2hex(right), " ", rk[i])


    combine = left + right


    cipher_text = permute(combine, final_perm, 64)
    return cipher_text
pt = "123456ABCD132536"
key = "AABB09182736CCDD"
key = hex2bin(key)
keyp = [57, 49, 41, 33, 25, 17, 9,
        1, 58, 50, 42, 34, 26, 18,
        10, 2, 59, 51, 43, 35, 27,
        19, 11, 3, 60, 52, 44, 36,
        63, 55, 47, 39, 31, 23, 15,
        7, 62, 54, 46, 38, 30, 22,
        14, 6, 61, 53, 45, 37, 29,
        21, 13, 5, 28, 20, 12, 4]
key = permute(key, keyp, 56)
shift_table = [1, 1, 2, 2,
               2, 2, 2, 2,
               1, 2, 2, 2,
              2, 2, 2, 1]
 key_comp = [14, 17, 11, 24, 1, 5,
             3, 28, 15, 6, 21, 10,
             23, 19, 12, 4, 26, 8,
             16, 7, 27, 20, 13, 2,
             41, 52, 31, 37, 47, 55,
             30, 40, 51, 45, 33, 48,
             44, 49, 39, 56, 34, 53,
             46, 42, 50, 36, 29, 32]

left = key[0:28]    # rkb for RoundKeys in binary
right = key[28:56]  # rk for RoundKeys in hexadecimal


rkb = []
rk = []
for i in range(0, 16):
    left = shift_left(left, shift_table[i])
    right = shift_left(right, shift_table[i])
    combine_str = left + right
    round_key = permute(combine_str, key_comp, 48)
    rkb.append(round_key)
    rk.append(bin2hex(round_key))
```

```
print("Name :- niraj italiya")

print ("190130107041 CE -A2")

print("Encryption")

cipher_text = bin2hex(encrypt(pt, rkb, rk))

print("Cipher Text : ", cipher_text)

print("Decryption")

rkb_rev = rkb[::-1]

rk_rev = rk[::-1]

text = bin2hex(encrypt(cipher_text, rkb_rev, rk_rev))

print("Plain Text : ", text)
```

output:-



```
Name :- niraj italiya
190130107041 CE -A2
Encryption
After initial permutation 14A7D67818CA18AD
Round  1   18CA18AD   5A78E394   194CD072DE8C
Round  2   5A78E394   4A1210F6   4568581ABCCE
Round  3   4A1210F6   B8089591   06EDA4ACF5B5
Round  4   B8089591   236779C2   DA2D032B6EE3
Round  5   236779C2   A15A4B87   69A629FEC913
Round  6   A15A4B87   2E8F9C65   C1948E87475E
Round  7   2E8F9C65   A9FC20A3   708AD2DDB3C0
Round  8   A9FC20A3   308BEE97   34F822F0C66D
Round  9   308BEE97   10AF9D37   84BB4473DCCC
Round  10   10AF9D37   6CA6CB20   02765708B5BF
Round  11   6CA6CB20   FF3C485F   6D5560AF7CA5
Round  12   FF3C485F   22A5963B   C2C1E96A4BF3
Round  13   22A5963B   387CCDAA   99C31397C91F
Round  14   387CCDAA   BD2DD2AB   251B8BC717D0
Round  15   BD2DD2AB   CF26B472   3330C5D9A36D
Round  16   19BA9212   CF26B472   181C5D75C66D
Cipher Text :  C0B7A8D05F3A829C
Decryption
After initial permutation 19BA9212CF26B472
Round  1   CF26B472   BD2DD2AB   181C5D75C66D
Round  2   BD2DD2AB   387CCDAA   3330C5D9A36D
Round  3   387CCDAA   22A5963B   251B8BC717D0
Round  4   22A5963B   FF3C485F   99C31397C91F
Round  5   FF3C485F   6CA6CB20   C2C1E96A4BF3
Round  6   6CA6CB20   10AF9D37   6D5560AF7CA5
Round  7   10AF9D37   308BEE97   02765708B5BF
Round  8   308BEE97   A9FC20A3   84BB4473DCCC
Round  9   A9FC20A3   2E8F9C65   34F822F0C66D
Round  10   2E8F9C65   A15A4B87   708AD2DDB3C0
Round  11   A15A4B87   236779C2   C1948E87475E
Round  12   236779C2   B8089591   69A629FEC913
Round  13   B8089591   4A1210F6   DA2D032B6EE3
Round  14   4A1210F6   5A78E394   06EDA4ACF5B5
Round  15   5A78E394   18CA18AD   4568581ABCCE
Round  16   14A7D678   18CA18AD   194CD072DE8C
Plain Text :  123456ABCD132536
```

Practical-6 : Implement key generation of AES.

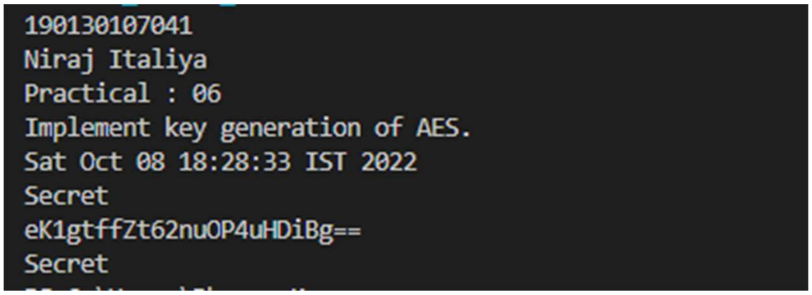
Code:

```
package Pra_06;
// package I;


import java.nio.charset.StandardCharsets;
import java.security.spec.KeySpec;
import java.util.Base64;
import java.util.Date;


import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.SecretKeySpec;


class AES {
    // Class private variables
    private static final String SECRET_KEY = "my_super_secret_key_ho_ho_ho";
    private static final String SALT = "ssshhhhhhhhhhh!!!!";
    // This method use to encrypt to string
    public static String encrypt(String strToEncrypt)
    {
        try {
```

```
        // Create default byte array

        byte[] iv = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

        IvParameterSpec ivspec = new IvParameterSpec(iv);


        // Create SecretKeyFactory object

        SecretKeyFactory factory = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");


        // Create KeySpec object and assign with

        // constructor

        KeySpec spec = new PBEKeySpec(SECRET_KEY.toCharArray(), SALT.getBytes(), 65536, 256);

        SecretKey tmp = factory.generateSecret(spec);

        SecretKeySpec secretKey = new SecretKeySpec(

        tmp.getEncoded(), "AES");


        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");

        cipher.init(Cipher.ENCRYPT_MODE, secretKey,ivspec);


        // Return encrypted string

        return Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes(StandardCharsets.UTF_8)));

    }


    catch (Exception e) {
          System.out.println("Error while encrypting: " + e.toString());
       }
       return null;
    }

    // This method use to decrypt to string
    public static String decrypt(String strToDecrypt)
    {
       try {

          // Default byte array
          byte[] iv = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

          // Create IvParameterSpec object and assign with
          // constructor
          IvParameterSpec ivspec = new IvParameterSpec(iv);

          // Create SecretKeyFactory Object
          SecretKeyFactory factory =
SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");

          // Create KeySpec object and assign with
          // constructor
          KeySpec spec = new PBEKeySpec( SECRET_KEY.toCharArray(), SALT.getBytes(),
65536, 256);
```

```
            SecretKey tmp = factory.generateSecret(spec);
            SecretKeySpec secretKey = new SecretKeySpec( tmp.getEncoded(), "AES");

            Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5PADDING");
            cipher.init(Cipher.DECRYPT_MODE, secretKey, ivspec);

            // Return decrypted string
            return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
        }
        catch (Exception e) {
            System.out.println("Error while decrypting: " + e.toString());
        }
        return null;
    }
}

// driver code
class Pra_06 {
    public static void main(String[] args)
    {
        System.out.println("190130107041");
        System.out.println("Niraj Italiya");
        System.out.println("Practical : 06");
        System.out.println("Implement key generation of AES.");
        Date date = new Date();
        System.out.println(date.toString());

        // Create String variables
        String originalString = "Secret";

        // Call encryption method
        String encryptedString = AES.encrypt(originalString);

        // Call decryption method
        String decryptedString = AES.decrypt(encryptedString);

        // Print all strings
        System.out.println(originalString);
        System.out.println(encryptedString);
        System.out.println(decryptedString);
    }
}
```

Program Execution Snapshot :



```
190130107041
Niraj Italiya
Practical : 06
Implement key generation of AES.
Sat Oct 08 18:28:33 IST 2022
Secret
eK1gtffZt62nuOP4uHDiBg==
Secret
```

Practical-7 :

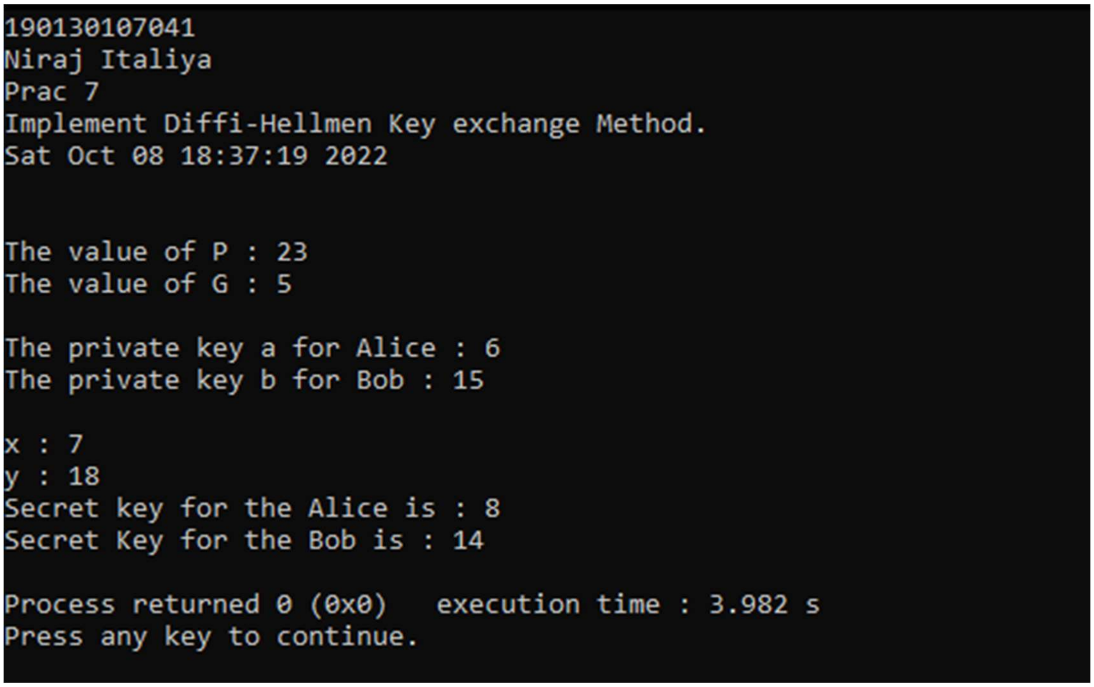
Code:

```
/* This program calculates the Key for two persons
using the Diffie-Hellman Key exchange algorithm */



#include<stdio.h>
#include<math.h>
#include<time.h>



// Power function to return value of a ^ b mod P
long long int power(long long int a, long long int b, long long int P)
{
    if (b == 1)
        return a;
    else
        return (((long long int)pow(a, b)) % P);


    // int r;
    // int y=1;


    // while(b>0){


    //     r=b%2;
    //     if (r==1){
    //         y=(y*a)%P;
    //     }
    //     a=a*a%P;
    //     b=b/2;
    // }


    // return y;


}


//Driver program
int main()
{
    long long int P, G, x, a, y, b, ka, kb;
    time_t t; // not a primitive datatype
```

```
time(&t);

printf("190130107041\n");
printf("Niraj Italiya\n");
printf("Prac 7\n");
printf("Implement Diffi-Hellmen Key exchange Method.\n");
printf("%s\n", ctime(&t));

// Both the persons will be agreed upon the
// public keys G and P

P = 23; // A prime number P is taken
printf("\nThe value of P : %lld\n", P);

G = 5; // A primitve root for P, G is taken
// G=9;
printf("The value of G : %lld\n\n", G);

// Alice will choose the private key a

a = 6; // a is the chosen private key
// a=4;
printf("The private key a for Alice : %lld\n", a);
x = power(G, a, P); // gets the generated key
```

```
    // Bob will choose the private key b
    b = 15; // b is the chosen private key
    // b=3;
    printf("The private key b for Bob : %lld\n\n", b);
    y = power(G, b, P); // gets the generated key

    printf("x : %lld\n", x);


    printf("y : %lld\n", y);
    // Generating the secret key after the exchange of keys
    ka = power(y, a, P); // Secret key for Alice
    printf("Secret key for the Alice is : %lld\n", ka);

    kb = power(x, b, P); // Secret key for Bob
    printf("Secret Key for the Bob is : %lld\n", kb);

    return 0;
}
```

Program Execution Snapshot :



```
190130107041
Niraj Italiya
Prac 7
Implement Diffi-Hellmen Key exchange Method.
Sat Oct 08 18:37:19 2022


The value of P : 23
The value of G : 5

The private key a for Alice : 6
The private key b for Bob : 15

x : 7
y : 18
Secret key for the Alice is : 8
Secret Key for the Bob is : 14

Process returned 0 (0x0)   execution time : 3.982 s
Press any key to continue.
```

Practical-8: Implement RSA key setup and encryption-decryption algorithm.

Code:

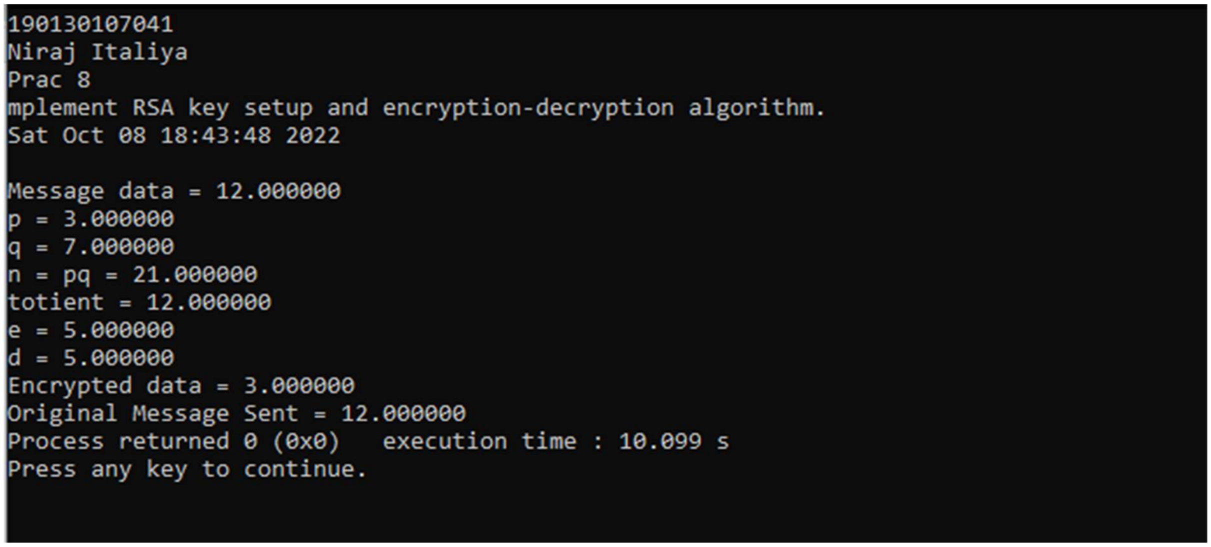
```
#include<stdio.h>
```

```
#include<math.h>
#include<time.h>

//to find gcd
int gcd(int a, int h)
{
    int temp;
    while(1)
    {
        temp = a%h;
        if(temp==0)
            return h;
        a = h;
        h = temp;
    }
}

int main()
{
    time_t t; // not a primitive datatype
    time(&t);

    printf("190130107041 \n");
    printf("Niraj Italiya\n");
    printf("Prac 8\n");
    printf("mplement RSA key setup and encryption-decryption algorithm.\n");


    printf("%s\n", ctime(&t));


    //2 random prime numbers
    double p = 3;
    double q = 7;


    double n=p*q;
    double count;
    double totient = (p-1)*(q-1);


    //public key
    //e stands for encrypt
    double e=2;


    //for checking co-prime which satisfies e>1
    while(e<totient){


        count = gcd(e,totient);
        if(count==1)
            break;
        else
            e++;
    }
```

```
//private key
//d stands for decrypt
double d;

//k can be any arbitrary value
double k = 2;

//choosing d such that it satisfies d*e = 1 + k * totient
d = (1 + (k*totient))/e;

double msg = 12;
double c = pow(msg,e);
double m = pow(c,d);

c=fmod(c,n);
m=fmod(m,n);

printf("Message data = %lf",msg);
printf("\np = %lf",p);
printf("\nq = %lf",q);
printf("\nn = pq = %lf",n);
printf("\ntotient = %lf",totient);
printf("\ne = %lf",e);
printf("\nd = %lf",d);
```

```
    printf("\nEncrypted data = %lf",c);

    printf("\nOriginal Message Sent = %lf",m);



    return 0;

}
```

Program Execution Snapshot:



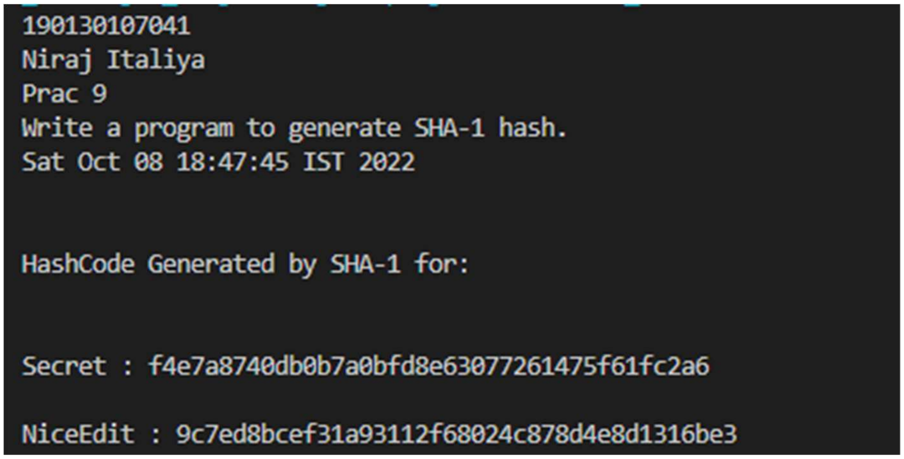
```
190130107041
Niraj Italiya
Prac 8
mplement RSA key setup and encryption-decryption algorithm.
Sat Oct 08 18:43:48 2022

Message data = 12.000000
p = 3.000000
q = 7.000000
n = pq = 21.000000
totient = 12.000000
e = 5.000000
d = 5.000000
Encrypted data = 3.000000
Original Message Sent = 12.000000
Process returned 0 (0x0)   execution time : 10.099 s
Press any key to continue.
```

Practical-9: Write a program to generate SHA-1 hash.

Code:

```
package Pra_09;

import java.math.BigInteger;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.*;


public class Pra9 {


    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

            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("190130107041");

        System.out.println("Niraj Italiya");

        System.out.println("Prac 9");

        System.out.println("Write a program to generate SHA-1 hash.");

        Date date = new Date();

        System.out.println(date.toString());


        System.out.println("\n");

        System.out.println("HashCode Generated by SHA-1 for: \n");


        String s1 = "Secret";

        System.out.println("\n" + s1 + " : " + encryptThisString(s1));


        String s2 = "NiceEdit";

        System.out.println("\n" + s2 + " : " + encryptThisString(s2));

    }

}
```

Program Execution Snapshot:



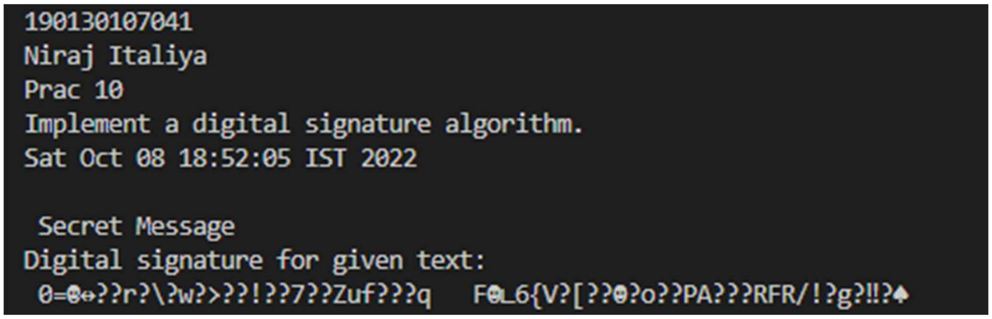
```
190130107041
Niraj Italiya
Prac 9
Write a program to generate SHA-1 hash.
Sat Oct 08 18:47:45 IST 2022


HashCode Generated by SHA-1 for:


Secret : f4e7a8740db0b7a0bfd8e63077261475f61fc2a6

NiceEdit : 9c7ed8bcef31a93112f68024c878d4e8d1316be3
```

Practical-10: Implement a digital signature algorithm.

Code:

```
package Pra_10;

package Pra_10;



import java.security.KeyPair;

import java.security.KeyPairGenerator;

import java.security.PrivateKey;

import java.security.Signature;
import java.util.*;
public class Pra_10 {
    public static void main(String args[]) throws Exception {

        //Accepting text from user
        System.out.println("190130107041");
        System.out.println("Niraj Italiya");
        System.out.println("Prac 10");
        System.out.println("Implement a digital signature algorithm.");

        Date date = new Date();
        System.out.println(date.toString());
        System.out.println("\n Secret Message");

        String msg = "Secret Message";

        //Creating KeyPair generator object
        KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("DSA");

        //Initializing the key pair generator
        keyPairGen.initialize(2048);

        //Generate the pair of keys
        KeyPair pair = keyPairGen.generateKeyPair();
        //Getting the private key from the key pair
        PrivateKey privKey = pair.getPrivate();

        //Creating a Signature object
        Signature sign = Signature.getInstance("SHA256withDSA");

        //Initialize the signature
        sign.initSign(privKey);
        byte[] bytes = msg.getBytes();

        //Adding data to the signature
        sign.update(bytes);

        //Calculating the signature
        byte[] signature = sign.sign();

        //Printing the signature
        System.out.println("Digital signature for given text:\n "+new String(signature, "UTF8"));
} }
```

Program Execution Snapshot:



```
190130107041
Niraj Italiya
Prac 10
Implement a digital signature algorithm.
Sat Oct 08 18:52:05 IST 2022

 Secret Message
Digital signature for given text:
 0=☻↔??r?\?w?>??!??7??Zuf???q   F☻∟6{V?[??☻?o??PA???RFR/!?g?‼?♠
```