SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMKUR-572103

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CRYPTOGRAPHY AND NETWORK SECURITY LABORATORY (7RCSL01)

Student Name: Khalid Farooq	USN: 1SI18CS046	Batch No: A3

Evaluation:

Write Up (10 marks)	**Clarity in concepts (10 marks)**	**Implementation and execution of the algorithms (10 marks)**	**Viva (05 marks)**	**Total (35 marks)**

Sl.No	Name of the Faculty In-Charge	Signature
1.	Sunitha N R	
2.	A H Shanthakumara	

Question No: 1

Perform encryption and decryption using mono-alphabetic cipher. The program should support the following:

i. Construct an input file named plaintext.txt (consisting of 1000 alphabets, without any space or special characters)

ii. Compute key space (Permutation of set of all letters appeared in plaintext.txt: there are n! permutations of a set of n elements)

iii. Encrypt the characters of plaintext.txt using any one key from (ii) and store the corresponding ciphertext characters in ciphertext.txt

iv. Compute the frequency of occurrence of each alphabet in both plaintext.txt and ciphertext.txt and tabulate the results as follows

Frequency	Plaintext character	Ciphertext character
12.34	A	X
.	.	.
.	.	.

2021-22

Monoalphabetic substitution cipher:

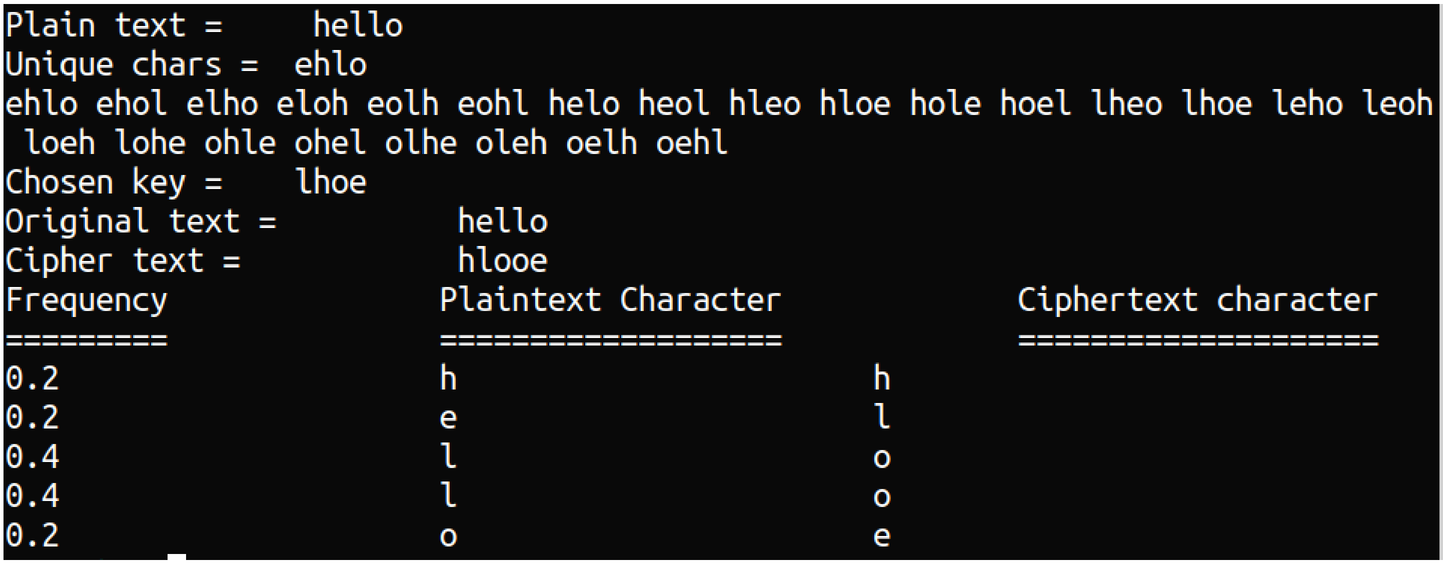
Select a Key randomly from 26! Key space and map from plain alphabet to cipher alphabet:

- Let us consider Plaintext P which contains every alphabets S = {a, b, c},
- There are 3! Permutations of S in a key space.
- Randomly chosen key K from key space.
- Map from plain alphabet to cipher alphabet

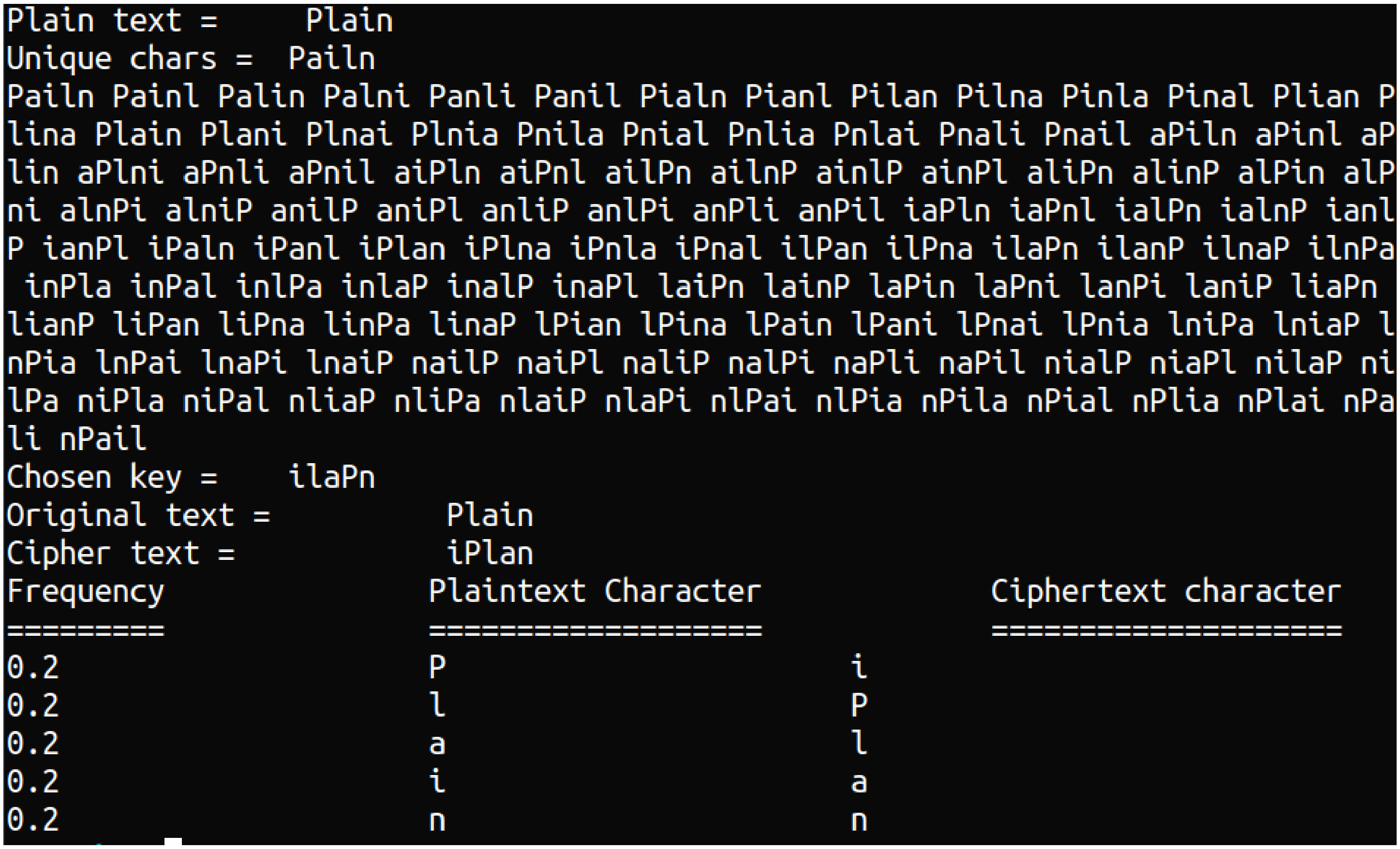
Program:

```
#include<bits/stdc++.h>

using namespace std ;

string uniquePlainText ;

string readPlainText(const char * name="plaintext.txt"){
    string pt ;
    ifstream fin;
    fin.open(name) ;
    fin>>pt ;
    return pt ;
}


void permute(string a, int l, int r , vector<string>& ks)
{
    if (l == r)
        ks.push_back(a) ;
    else
    {
        for (int i = l; i <= r; i++)
        {
            swap(a[l], a[i]);

            permute(a, l+1, r , ks);
            swap(a[l], a[i]);
        }
    }
}

vector<string> genKeySpace(string pt ){
    set<char> charset ;
    for(int i =0 ;i < pt.length() ; i++) charset.insert(pt[i]) ;

    uniquePlainText  = string(charset.begin() , charset.end()) ;

    vector<string> keyspace ;
    permute(uniquePlainText , 0 , uniquePlainText.length()-1 , keyspace )
;
    return keyspace ;
}


string encryptUsingKey(string uniq , string key){
    string pt = readPlainText() ;
    string ct = "" ;
    for(int i =0 ;i < pt.length() ; i++){
         char c = pt[i] ;
         ct+=(key[uniq.find(c)]) ;
    }
    cout<<"Original text = \t " << pt << endl;
    cout<< "Cipher text = \t\t " << ct <<endl;
    return ct ;
}

void saveToFile(string data , string filename ="ciphertext.txt" ){
    ofstream fout ;
    fout.open(filename.c_str()) ;
    fout<< data ;
    fout.close() ;
}


void showFrequency(string pt , string ct){
    map<char , char > mPlain ;
    map<char , char > mCipher ;

    for(int i =0 ;i < pt.length() ; i++){
         mPlain[pt[i]]++ ;
         mCipher[ct[i]]++ ;
    }
    cout<<"Frequency\t\tPlaintext Character\t\tCiphertext character"
<<endl;
    cout<<"=========\t\t===================\t\t===================="
<<endl;
    for(int i =0 ;i < pt.length() ; i++){
         cout<< (float)mPlain[pt[i]]/pt.length() << "\t\t\t" << pt[i] <<
"\t\t\t" << ct[i] << endl ;
    }

 }


 int main(void){
    srand(time(0)) ;
    string pt = readPlainText() ;
    cout<<"Plain text = \t " << pt << endl;
    vector<string> keyspace = genKeySpace(pt) ;
    string key = keyspace[rand()%keyspace.size()] ;

    cout<<"Unique chars = \t" << uniquePlainText <<endl;

    for(int i=0; i < keyspace.size(); i++)
         std::cout << keyspace.at(i) << ' ';
    cout<<endl;
    cout<<"Chosen key = \t" << key <<endl;
    string ct = encryptUsingKey(uniquePlainText , key) ;
    saveToFile(ct) ;
    showFrequency(pt , ct) ;
 }
```

OUTPUT :



```
Plain text =     hello
Unique chars =  ehlo
ehlo ehol elho eloh eolh eohl helo heol hleo hloe hole hoel lheo lhoe leho leoh
 loeh lohe ohle ohel olhe oleh oelh oehl
Chosen key =    lhoe
Original text =          hello
Cipher text =            hlooe
Frequency               Plaintext Character             Ciphertext character
=========               ===================             ====================
0.2                     h                       h
0.2                     e                       l
0.4                     l                       o
0.4                     l                       o
0.2                     o                       e
```



```
Plain text =     Plain
Unique chars =  Pailn
Pailn Painl Palin Palni Panli Panil Pialn Pianl Pilan Pilna Pinla Pinal Plian P
lina Plain Plani Plnai Plnia Pnila Pnial Pnlia Pnlai Pnali Pnail aPiln aPinl aP
lin aPlni aPnli aPnil aiPln aiPnl ailPn ailnP ainlP ainPl aliPn alinP alPin alP
ni alnPi alniP anilP aniPl anliP anlPi anPli anPil iaPln iaPnl ialPn ialnP ianl
P ianPl iPaln iPanl iPlan iPlna iPnla iPnal ilPan ilPna ilaPn ilanP ilnaP ilnPa
 inPla inPal inlPa inlaP inalP inaPl laiPn lainP laPin laPni lanPi laniP liaPn
lianP liPan liPna linPa linaP lPian lPina lPain lPani lPnai lPnia lniPa lniaP l
nPia lnPai lnaPi lnaiP nailP naiPl naliP nalPi naPli naPil nialP niaPl nilaP ni
lPa niPla niPal nliaP nliPa nlaiP nlaPi nlPai nlPia nPila nPial nPlia nPlai nPa
li nPail
Chosen key =    ilaPn
Original text =          Plain
Cipher text =            iPlan
Frequency               Plaintext Character             Ciphertext character
=========               ===================             ====================
0.2                     P                       i
0.2                     l                       P
0.2                     a                       l
0.2                     i                       a
0.2                     n                       n
```

Student Name: Khalid Farooq	USN: 1SI18CS046	Batch No: A3	Date:

Evaluation:

Write Up (10 marks)	**Clarity in concepts (10 marks)**	**Implementation and execution of the algorithms (10 marks)**	**Viva (05 marks)**	**Total (35 marks)**

Sl.No	Name of the Faculty In-Charge	Signature
1.	Sunitha N R	
2.	A H Shanthakumara	

Question No: 2

Write a program to perform the following using Playfair cipher technique

(i) Encrypt a given message M with different keys {k_1, k_2,…,k_n}. Print key and ciphertext pair

(ii) Decrypt the cipher texts obtained in (i) to get back M

Playfair Cipher:

Construct a 5 X 5 matrix using a keyword from left to right and from top to bottom, and then filling in the remainder of the matrix with the remaining letters in alphabetical order. The letters I and J count as one letter.

Plaintext is encrypted two letters at a time, according to the following rules:

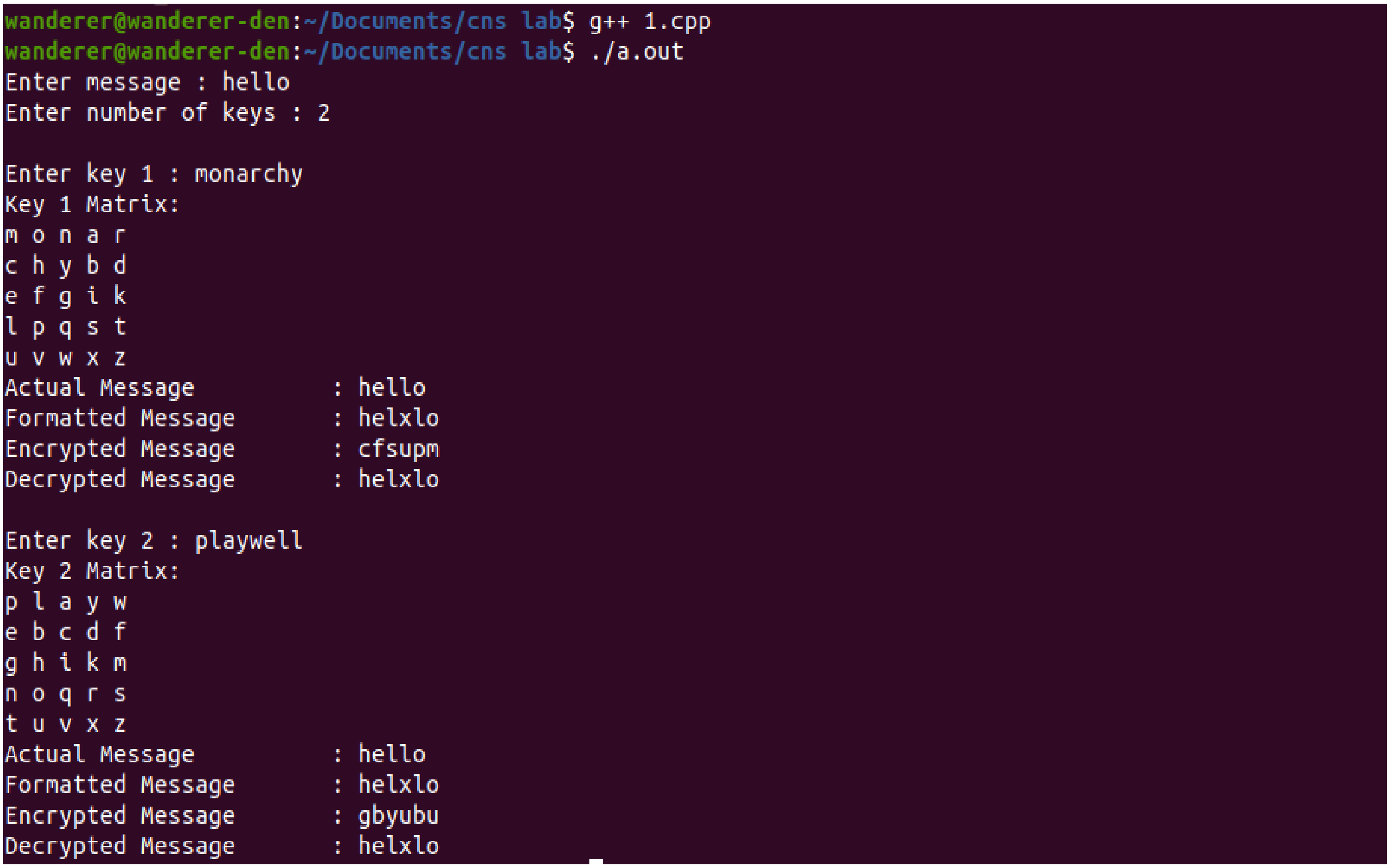
1. Repeating plaintext letters that are in the same pair are separated with a filler letter, such as x, so that balloon would be treated as ba lx lo on.
2. Two plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row circularly following the last.
3. Two plaintext letters that fall in the same column are each replaced by the letter beneath, with the top element of the column circularly following the last.
4. Otherwise, each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter.

Program:

```
#include <bits/stdc++.h>
using namespace std;

typedef struct
{
     int row;
     int col;
} position;

char mat[5][5];

void generatematrix (string key)
{
     int flag[26] = { 0 }; int x = 0, y = 0;
     for (int i = 0; i < key.length (); i++)
     {
          if (key[i] == 'j')
               key[i] = 'i';
          if (flag[key[i] - 'a'] == 0)
          {
               mat[x][y++] = key[i];
               flag[key[i] - 'a'] = 1;
          }
          if (y == 5)
          {
               x++;
               y = 0;
          }
     }
     for (char ch = 'a'; ch <= 'z'; ch++)
     {
          if (ch == 'j') continue;
          if (flag[ch - 'a'] == 0)
          {
               mat[x][y++] = ch;
               flag[ch - 'a'] = 1;
          }
          if (y == 5)
          {
          x++; y = 0;
          }
     }
}
```

```
string formatmessage (string msg)
{
     for (int i = 0; i < msg.length (); i++) if (msg[i] == 'j')
     msg[i] = 'i';
     for (int i = 1; i < msg.length (); i += 2) if (msg[i - 1] == msg[i])
     msg.insert (i, "x");
     if (msg.length () % 2 != 0) msg += 'x';
     return msg;
}

position getposition (char c)
{
     position p;
     for (int i = 0; i < 5; i++)
     {
          for (int j = 0; j < 5; j++)
          {
               if (c == mat[i][j])
               {
                    p ={i, j};
                    return p;
               }
          }
     }
     return p;
}

string encrypt (string message)
{
     string ctext;
     for (int i = 0; i < message.length (); i += 2)
     {
          position p1 = getposition (message[i]); position p2 =
getposition (message[i + 1]); int x1 = p1.row;
          int y1 = p1.col; int x2 = p2.row; int y2 = p2.col; if (x1 == x2)
          {
               ctext.append (1, mat[x1][(y1 + 1) % 5]);
               ctext.append (1, mat[x2][(y2 + 1) % 5]);
          }
          else if (y1 == y2)
          {
               ctext.append (1, mat[(x1 + 1) % 5][y1]);
               ctext.append (1, mat[(x2 + 1) % 5][y2]);
          }
          else
          {
               ctext.append (1, mat[x1][y2]);
               ctext.append (1, mat[x2][y1]);
          }
     }

     return ctext;
}

string decrypt (string message)
{
     string ptext;
     for (int i = 0; i < message.length (); i += 2)
     {
          position p1 = getposition (message[i]); position p2 =
getposition (message[i + 1]); int x1 = p1.row;
          int y1 = p1.col; int x2 = p2.row; int y2 = p2.col;
          if (x1 == x2)
          {
               ptext.append (1, mat[x1][--y1 < 0 ? 4 : y1]);
               ptext.append (1, mat[x2][--y2 < 0 ? 4 : y2]);
          }
          else if (y1 == y2)
          {
               ptext.append (1, mat[--x1 < 0 ? 4 : x1][y1]);
               ptext.append (1, mat[--x2 < 0 ? 4 : x2][y2]);
          }
          else{
               ptext.append (1, mat[x1][y2]);
               ptext.append (1, mat[x2][y1]);
               }
          }

     return ptext;
}

int main ()
{
     string plaintext;
     cout << "Enter message:"; cin >> plaintext;
     int n;
     cout << "Enter number of keys:"; cin >> n;
     string key[n];
     for (int i = 0; i < n; i++)
     {
          cout << "\nEnter key" << i + 1 << ":"; cin >> key[i];
          generatematrix (key[i]);
          cout << "\nKey" << i + 1 << "Matrix" << endl; for (int k = 0; k
< 5; k++)
          {
               for (int j = 0; j < 5; j++)
               {
                    cout << mat[k][j] << " ";
               }
               cout << endl;
          }
          cout << "Actual message: " << plaintext << endl; string fmsg =
formatmessage (plaintext);
          cout << "Formatted message: " << fmsg << endl; string ciphertext
= encrypt (fmsg);
          cout << "Encrypted message: " << ciphertext << endl; string
decryptmessage = decrypt (ciphertext);
          cout << "Decrypted message: " << decryptmessage << endl;
     }
     return 0;
}
```

Output:



```
wanderer@wanderer-den:~/Documents/cns lab$ g++ 1.cpp
wanderer@wanderer-den:~/Documents/cns lab$ ./a.out
Enter message : hello
Enter number of keys : 2

Enter key 1 : monarchy
Key 1 Matrix:
m o n a r
c h y b d
e f g i k
l p q s t
u v w x z
Actual Message          : hello
Formatted Message       : helxlo
Encrypted Message       : cfsupm
Decrypted Message       : helxlo

Enter key 2 : playwell
Key 2 Matrix:
p l a y w
e b c d f
g h i k m
n o q r s
t u v x z
Actual Message          : hello
Formatted Message       : helxlo
Encrypted Message       : gbyubu
Decrypted Message       : helxlo
```

Siddaganga Institute of Technology, Tumkur-572103

Department of Computer Science and Engineering

CRYPTOGRAPHY AND NETWORK SECURITY LAB (7RCSL01)

Student Name: Khalid Faroooq	USN: 1SI18CS046	Batch No: A3	Date:

Evaluation:

Write Up (10 marks)	**Clarity in concepts (10 marks)**	**Implementation and execution of the algorithms (10 marks)**	**Viva (05 marks)**	**Total (35 marks)**

Sl.No	Name of the Faculty In-Charge	Signature
1.	Sunitha N R	
2.	A H Shanthakumara	

Question No: 3

Write a program to perform the following using Hill cipher:

(i) Encrypt a message M with a given key matrix of size 2X2 and 3X3

(i) Decrypt the cipher text obtained in (i) by computing inverse of the respective key matrix

Hill Cipher:

This encryption algorithm takes m successive plaintext letters and substitutes for them m ciphertext letters. The substitution is determined by m linear equations in which each character is assigned a numerical value (a = 0, b = 1, , z = 25) . For m = 3, the system can be described as

$c_1 = (k_{11}p_1 + k_{12}p_2 + k_{13}p_3) \mathrm{mod}\ 26$

$c_2 = (k_{21}p_1 + k_{22}p_2 + k_{23}p_3) \mathrm{mod}\ 26$

$c_3 = (k_{31}p_1 + k_{32}p_2 + k_{33}p_3) \mathrm{mod}\ 26$

C = PK mod 26 where C and P are row vectors of length 3 representing the plaintext and ciphertext, and K is a 3 X 3 matrix representing the encryption key. Operations are performed mod 26.

Decryption requires using the inverse of the matrix K.

C = E(K, P) = PK mod 26

$P = D(K, C) = CK^{-1} \mathrm{mod}\ 26 = PKK^{-1} = P$

For the 2X2 matrix determinant is $k_{11}k_{22} - k_{12}k_{21}$. For a 3X3 matrix, the value of the determinant is $k_{11}k_{22}k_{33} + k_{21}k_{32}k_{13} + k_{31}k_{12}k_{23} - k_{31}k_{22}k_{13} - k_{21}k_{12}k_{33} - k_{11}k_{32}k_{23}$

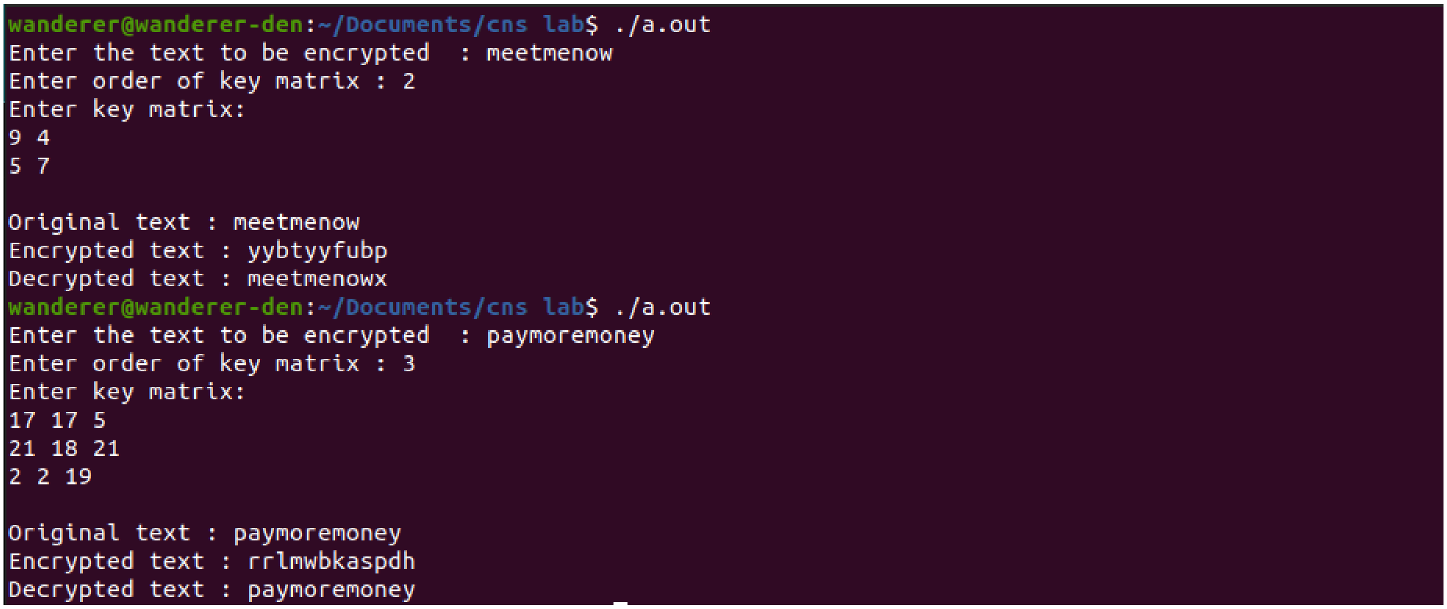
If a square matrix A has a nonzero determinant, then the inverse of the matrix is computed as $[A^{-1}]_{ij} = (\det A)^{-1}(-1)^{i+j}(D_{ji})$, where (D_{ji}) is the sub determinant formed by deleting the ‘j’th row and the’ i’th column of A, det(A) is the determinant of A, and $(\det A)^{-1}$ is the multiplicative inverse of (det A) mod 26.

Program:

```
#include<bits/stdc++.h>
using namespace std ;

int key[3][3] ;

int mod26(int x)
{
    return x >= 0 ? (x%26) : 26-(abs(x)%26) ;
}

int findDet(int m[3][3] , int n)
{
    int det;
    if(n == 2)
    {
      det = m[0][0] * m[1][1] - m[0][1]*m[1][0] ;
    }
    else if (n == 3)
    {
      det = m[0][0]*(m[1][1]*m[2][2] - m[1][2]*m[2][1]) -
m[0][1]*(m[1][0]*m[2][2] - m[2][0]*m[1][2] ) + m[0][2]*(m[1][0]*m[2][1] -
m[1][1]*m[2][0]);
    }
    else det = 0 ;

    return mod26(det);
}

int findDetInverse(int R , int D = 26)
{
    int i = 0 ;
    int p[100] = {0,1};
    int q[100] = {0} ;

    while(R!=0)
    {
      q[i] = D/R ;
      int oldD = D ;
      D = R ;
      R = oldD%R ;

      if(i>1)
      {
           p[i] = mod26(p[i-2] - p[i-1]*q[i-2]) ;
      }
      i++ ;
    }

    if (i == 1) return 1;
    else return p[i] = mod26(p[i-2] - p[i-1]*q[i-2]) ;
```

```
}

void multiplyMatrices(int a[1000][3] , int a_rows , int a_cols , int b[1000][3]
, int b_rows , int b_cols , int res[1000][3])
{
    for(int i=0 ; i < a_rows ; i++)
    {
      for(int j=0 ; j < b_cols ; j++)
      {
            for(int k=0 ; k < b_rows ; k++)
            {
                 res[i][j] += a[i][k]*b[k][j];
            }
            res[i][j] = mod26(res[i][j]);
      }
    }
}

void findInverse(int m[3][3] , int n , int m_inverse[3][3] )
{
    int adj[3][3] = {0};

    int det = findDet(m , n);
    int detInverse = findDetInverse(det);

    if(n==2)
    {
      adj[0][0] = m[1][1];
      adj[1][1] = m[0][0];
      adj[0][1] = -m[0][1];
      adj[1][0] = -m[1][0];
    }
    else if(n==3)
    {
      int temp[5][5] = {0} ;

      for(int i=0; i<5; i++)
      {
            for(int j=0; j<5; j++)
            {
                 temp[i][j] = m[i%3][j%3] ;
            }
      }

      for(int i=1; i<=3 ; i++)
      {
            for(int j=1; j<=3 ; j++)
            {
                 adj[j-1][i-1] = temp[i][j]*temp[i+1][j+1] -
temp[i][j+1]*temp[i+1][j];
            }
      }
    }
    for(int i=0; i<n ; i++)
    {
      for(int j=0; j<n ; j++)
      {
            m_inverse[i][j] = mod26(adj[i][j] * detInverse) ;
      }
    }
}

string encrypt(string pt, int n)
{
    int P[1000][3] = {0} ;
    int C[1000][3] = {0} ;
    int ptIter = 0 ;

    while(pt.length()%n != 0)
    {
      pt += "x" ;
    }

    int row = (pt.length())/n;
    for(int i=0; i<row ; i++)
    {
      for(int j=0; j<n; j++)
      {
            P[i][j] = pt[ptIter++]-'a' ;
      }
    }

    multiplyMatrices(P, row , n , key , n , n , C) ;

    string ct = "" ;
    for(int i=0 ; i<row ; i++)
    {
      for(int j=0 ; j<n ;j++)
      {
            ct += (C[i][j] + 'a');
      }
    }
    return ct ;
}

string decrypt(string ct, int n)
{
    int P[1000][3] = {0} ;
    int C[1000][3] = {0} ;
    int ctIter = 0 ;

    int row = ct.length()/n;

    for(int i=0; i<row ; i++)
    {
      for(int j=0; j<n; j++)
      {
            C[i][j] = ct[ctIter++]-'a' ;
      }
    }

    int k_inverse[3][3] = {0};

    findInverse(key, n , k_inverse);
    multiplyMatrices(C, row , n , k_inverse , n , n , P) ;

    string pt = "" ;
    for(int i = 0 ; i<row ; i++)
    {
      for(int j=0 ; j<n ; j++)
      {
            pt += (P[i][j] + 'a');
      }
    }
    return pt ;
}

int main(void)
{
    string pt ;
    int n ;

    cout << "Enter the text to be encrypted    : " ;
    getline(cin,pt);

    cout << "Enter order of key matrix : ";
    cin >> n ;

    pt.erase(remove(pt.begin(), pt.end(), ' '), pt.end());

    cout<<"Enter key matrix: " <<endl;
    for(int i=0; i<n; i++)
    {
      for(int j=0; j<n; j++)
      {
            cin >> key[i][j];
      }
    }

    cout << "\nOriginal text : " << pt << endl;

    string ct = encrypt(pt, n) ;
    cout << "Encrypted text : " << ct << endl;

    string dt = decrypt(ct, n);
    cout << "Decrypted text : " << dt << endl;
}
```

OUTPUT :



```
wanderer@wanderer-den:~/Documents/cns lab$ ./a.out
Enter the text to be encrypted  : meetmenow
Enter order of key matrix : 2
Enter key matrix:
9 4
5 7

Original text : meetmenow
Encrypted text : yybtyyfubp
Decrypted text : meetmenowx
wanderer@wanderer-den:~/Documents/cns lab$ ./a.out
Enter the text to be encrypted  : paymoremoney
Enter order of key matrix : 3
Enter key matrix:
17 17 5
21 18 21
2 2 19

Original text : paymoremoney
Encrypted text : rrlmwbkaspdh
Decrypted text : paymoremoney
```

SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMKUR-572103

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CRYPTOGRAPHY AND NETWORK SECURITY LAB (7RCSL01)

Student Name: Khalid Farooq	USN: 1SI18CS046	Batch No: A3	Date:

Evaluation:

Write Up (10 marks)	**Clarity in concepts (10 marks)**	**Implementation and execution of the algorithms (10 marks)**	**Viva (05 marks)**	**Total (35 marks)**

Sl.No	Name of the Faculty In-Charge	Signature
1.	Sunitha N R	
2.	A H Shanthakumara	

Question No: 4

Write a program to perform encryption and decryption using transposition technique with column permutation given as key.

Transposition technique:

Write the message in a rectangle, row by row, and read the message off, column by column, but permute the order of the columns. The order of the columns then becomes the key to the algorithm.

Example:

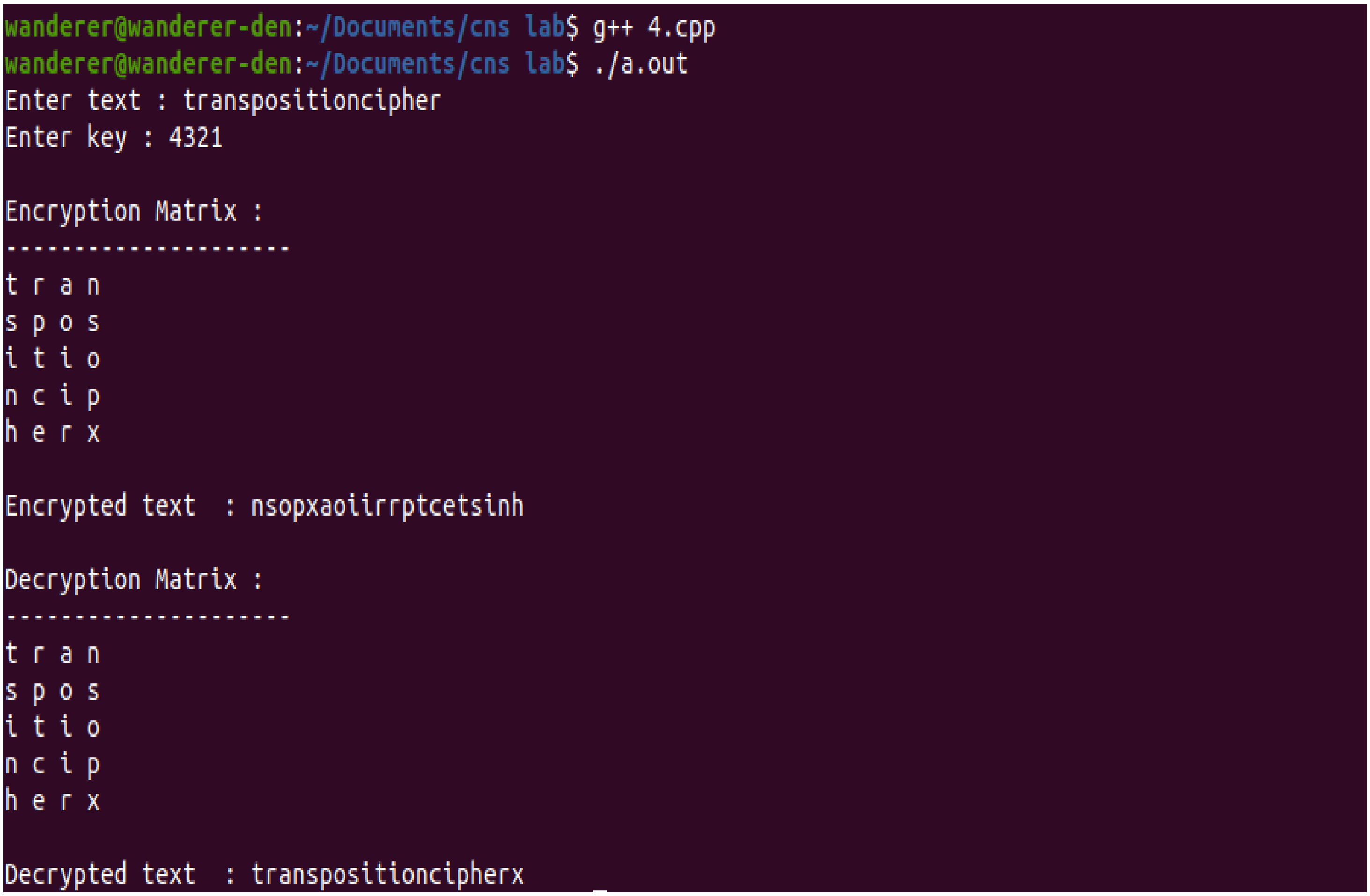
```
Key:           4 3 1 2 5 6 7
Plain text:    a t t a c k p
               o s t p o n e
               d u n t i l t
               w o a m x y z
Cipher text: TTNAAPTMTSUOAODWCOIXKNLYPETZ
```

Let us consider the key is 4312567. To encrypt, start with the column that is labeled 1, in this case column 3. Write down all the letters in that column. Proceed to column 4, which is labeled 2, then column 2, then column 1, then columns 5, 6, and 7.

Program:

```
#include<bits/stdc++.h>
using namespace std ;

string encrypt(string pt , string key)
{
    string ct = "";
    int k = 0;

    int num_row = ceil((float) pt.length()/key.length());
    int num_col = key.length();
    char mat[num_row][num_col];

    cout << "\nEncryption Matrix :" << endl;
    cout << "---------------------" << endl;
    for(int i=0; i<num_row ; i++)
    {
      for(int j=0; j<num_col; j++)
      {
           if(k < pt.length())
           {
                cout << (mat[i][j] = pt[k++]) << " ";
           }
           else
           {
                cout << (mat[i][j] = 'x') << " " ;
           }
      }
      cout << endl;
    }

    for(int i=0; i<num_col; i++)
    {
      for(int j=0; j<num_row; j++)
      {
           ct += mat[j][key.find(i+'1')];
      }
    }
    return ct;
}

string decrypt(string ct , string key)
{
    string pt = "";
    int k = 0;

    int num_row = ceil((float)ct.length() / key.length());
    int num_col = key.length();
    char mat[num_row][num_col];
    for(int i=0; i<num_col; i++)
    {
      for(int j=0; j<num_row; j++)
      {
           mat[j][key.find(i+'1')] = ct[k++];
      }
    }

    cout << "\nDecryption Matrix :" << endl;
    cout << "---------------------" << endl;

    for(int i=0; i<num_row ; i++)
    {
      for(int j=0; j<num_col; j++)
      {
           cout << mat[i][j] << " ";
           pt += mat[i][j];
      }
      cout << endl;
    }
    return pt;
}

int main()
{
    string plaintext , key , ciphertext , decryptext;

    cout << "Enter text : ";
    getline(cin,plaintext);

    cout << "Enter key : ";
    getline(cin,key);

    plaintext.erase(remove(plaintext.begin(), plaintext.end(), ' '), 
plaintext.end());

    ciphertext = encrypt(plaintext , key);
    cout << "\nEncrypted text \t: " << ciphertext << endl;

    decryptext = decrypt(ciphertext , key);
    cout << "\nDecrypted text \t: " << decryptext << endl;
}
```

Output:



```
wanderer@wanderer-den:~/Documents/cns lab$ g++ 4.cpp
wanderer@wanderer-den:~/Documents/cns lab$ ./a.out
Enter text : transpositioncipher
Enter key : 4321

Encryption Matrix :
---------------------
t r a n
s p o s
i t i o
n c i p
h e r x

Encrypted text  : nsopxaoiirrptcetsinh

Decryption Matrix :
---------------------
t r a n
s p o s
i t i o
n c i p
h e r x

Decrypted text  : transpositioncipherx
```

SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMKUR-572103

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CRYPTOGRAPHY AND NETWORK SECURITY LAB (7RCSL01)

Student Name: Khalid Faroooq	USN: 1SI18CS046	Batch No: A3	Date:

Evaluation:

Write Up (10 marks)	Clarity in concepts (10 marks)	Implementation and execution of the algorithms (10 marks)	Viva (05 marks)	Total (35 marks)

Sl.No	Name of the Faculty In-Charge	Signature
1.	Sunitha N R	
2.	A H Shanthakumara	

Question No: 5

Generate and print 48-bit keys for all sixteen rounds of DES algorithm, given a 64-bit initial key.

Algorithm: To Generate 48-bits key, follow the flow-chart and tables given below.

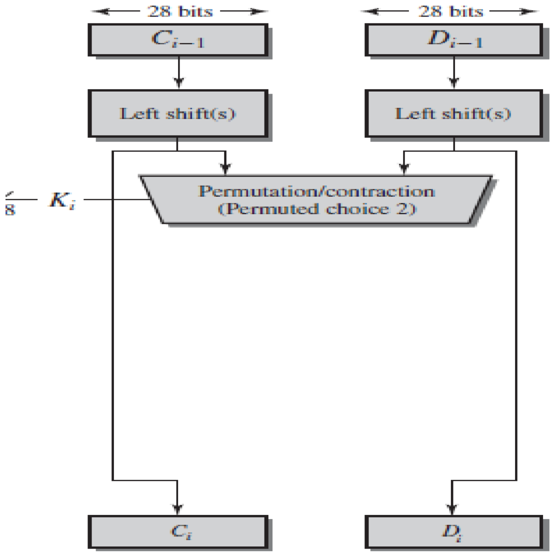
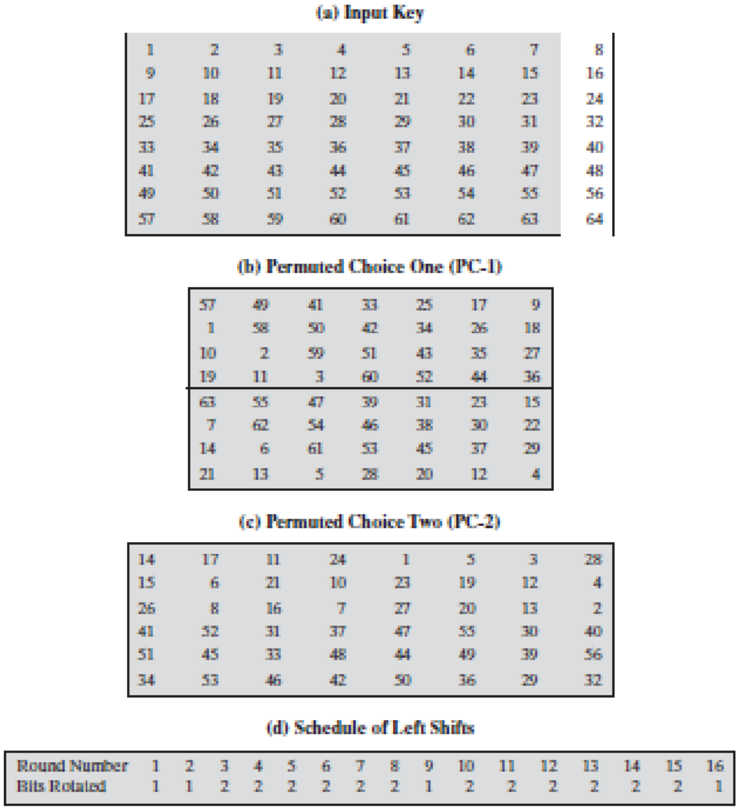


Figure: DES key Schedule Calculation



(a) Input Key

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

(b) Permuted Choice One (PC-1)

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

(c) Permuted Choice Two (PC-2)

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

(d) Schedule of Left Shifts

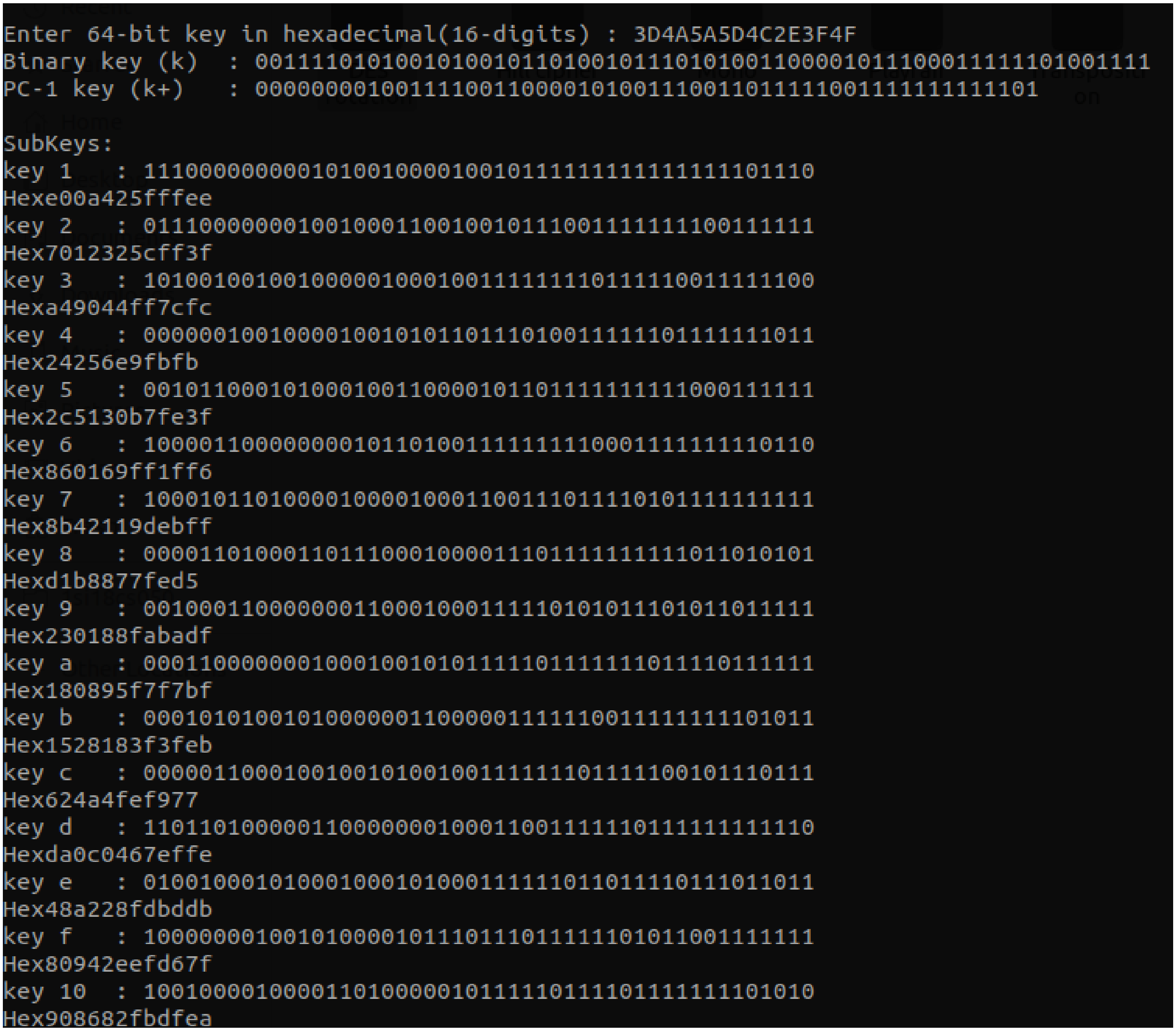
Round Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits Rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Tables: DES key Schedule Calculation

PROGRAM:

```
#include <bits/stdc++.h>
#include <cstring>
using namespace std;

int permChoiceOne[] = {
                            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 };

int permChoiceTwo[] = {
                            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 };

int leftShiftTable[] = {1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1};

string rotateSubKey(string s , int rot)
{
    return s.substr(rot, s.length()-rot) + s.substr(0, rot) ;
}

string firstPermute(string input)
{
    string res = "" ;
    for(int i=0 ; i<56 ; i++)
    {
      res += input[permChoiceOne[i]-1];
    }
    return res ;
}

string secondPermute(string input)
{
    string res = "" ;
    for(int i=0 ; i<48 ; i++)
    {
      res += input[permChoiceTwo[i]-1];
    }
    return res ;
}

void genKeys(string left, string right)
{
    ofstream fout ;
    fout.open("keygen.txt");
    for (int i=0; i<16; i++)
    {
      left = rotateSubKey(left , leftShiftTable[i]);
      right = rotateSubKey(right, leftShiftTable[i]);
      string key = secondPermute(left+right);
      cout << "key " << i+1 << " \t: " << key << endl;
      unsigned long long res= bitset<48>(key).to_ulong();
      cout<<"Hex"<<hex<<res<<endl;
      fout << key << endl;
    }
}

int main()
{
    unsigned long long hexkey;
    cout << "\nEnter 64-bit key in hexadecimal(16-digits) : " ;
    cin >> hex >> hexkey;

    string key = bitset<64>(hexkey).to_string();
    cout << "Binary key (k) \t: " << key << endl;

    key = firstPermute(key) ;
    cout << "PC-1 key (k+) \t: " << key << endl;

    cout << "\nSubKeys: " << endl;
    genKeys(key.substr(0,28) , key.substr(28,28));

    cout<<endl<<endl ;

    return 0;
}
```

OUTPUT:



```
Enter 64-bit key in hexadecimal(16-digits) : 3D4A5A5D4C2E3F4F
Binary key (k)  : 0011110101001010010110100101110101001100001011100011111101001111
PC-1 key (k+)   : 00000000100111100110000101001110011011111001111111111101

SubKeys:
key 1   : 111000000000101001000010010111111111111111101110
Hexe00a425fffee
key 2   : 011100000001001000110010010111001111111100111111
Hex7012325cff3f
key 3   : 101001001001000001000100111111110111110011111100
Hexa49044ff7cfc
key 4   : 000000100100001001010110111010011111101111111011
Hex24256e9fbfb
key 5   : 001011000101000100110000101101111111111000111111
Hex2c5130b7fe3f
key 6   : 100001100000000101101001111111110001111111110110
Hex860169ff1ff6
key 7   : 100010110100001000010001100111011110101111111111
Hex8b42119debff
key 8   : 000011010001101110001000011101111111111011010101
Hexd1b8877fed5
key 9   : 001000110000000110001000111110101011101011011111
Hex230188fabadf
key a   : 000110000000100010010101111101111111011110111111
Hex180895f7f7bf
key b   : 000101010010100000011000001111110011111111101011
Hex1528183f3feb
key c   : 000001100010010010100100111111101111100101110111
Hex624a4fef977
key d   : 110110100000110000000100011001111110111111111110
Hexda0c0467effe
key e   : 010010001010001000101000111111011011110111011011
Hex48a228fdbddb
key f   : 100000001001010000101110111011111101011001111111
Hex80942eefd67f
key 10  : 100100001000011010000010111110111101111111101010
Hex908682fbdfea
```