SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMKUR-572103

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CRYPTOGRAPHY AND NETWORK SECURITY LABORATORY (7RCSL01)

Student Name: ANVS Anudeep	**USN: 1SI19CS017**	**Batch No:A1**	**Date: 21/11/2022**

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.	Dr AS Poornima	
2.	Ravi V	

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
.	.	.
.	.	.

Monoalphabetic substitution cipher:

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

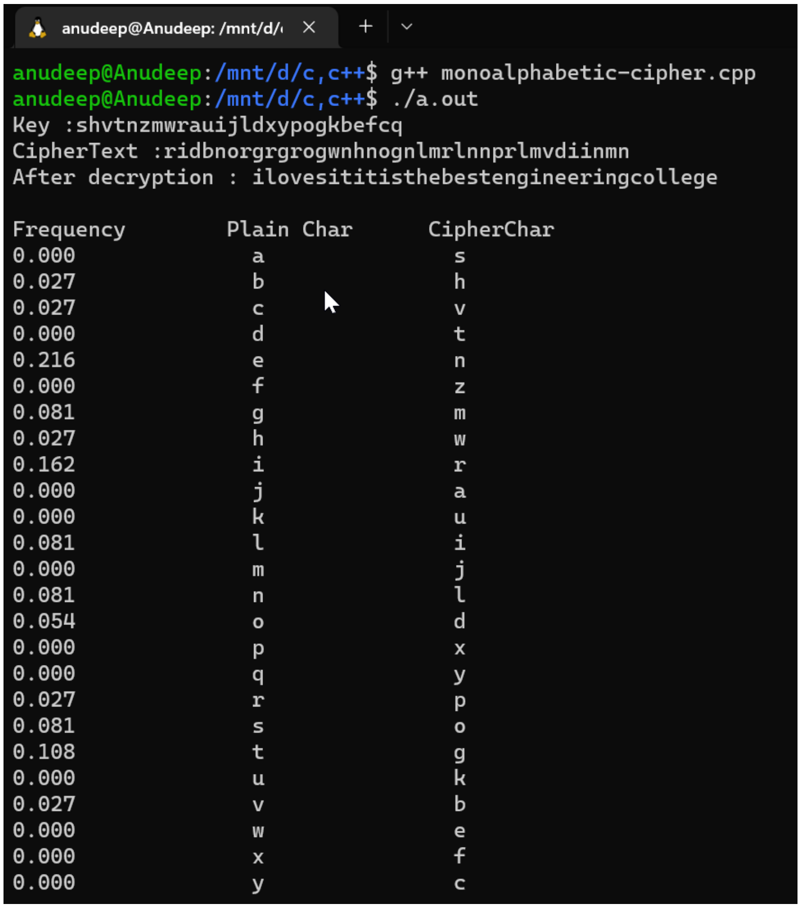
- Consider Plaintext P which contains sequence of characters.
- Consider the alphabet { a,b,c……..z}
- Consider a Initial Key which also contains only alphabets K= {a,b, …… z} to have the keyspace.
- Hence there will be 26! Keyspace.
- User has to generate randomly any one possible permutation of alphabets {a…z} say key K .
- Define each letter in the randomly generated key K against each letter of the plain text.
- Map from plain alphabet to cipher alphabet

```
#include <bits/stdc++.h>
using namespace std;
string GenKey() // generate key
{
    srand(time(NULL));
    string key = "";
    int t = rand();
    int p = 0;
    char temp;
    char c;
    // Store all unique characters
    for (int i = 0; i < 26; i++)
    {
        c = 'a';
        c = (char)(c + i);
        key += c;
    }
    // Shuffle characters to generate key
    while (p < 26)
    {
        t = p + rand() % (26 - p);
        temp = key[p];
        key[p] = key[t];
        key[t] = temp;
        p++;
    }
    cout << "Key :" << key << endl;
    return key;
}
void MapKey(unordered_map<char, char> &enKeyMap, unordered_map<char, char> &deKeyMap, string key)
{
    // Map the keys to a character in the alphabet
    char c;
    for (int i = 0; i < 26; i++)
    {
```

```
 c = 'a';
        c = (char)(c + i);
        enKeyMap[c] = key[i];
        deKeyMap[key[i]] = c;
    }
}
void Encrypt(string &pText, string &cText, unordered_map<char, char> enKeyMap)
{
    // Encrypt the plain text from file
    for (int i = 0; i < pText.length(); i++)
    {
        cText += enKeyMap[pText[i]];
    }
}

void Decrypt(string &cText, unordered_map<char, char> deKeyMap)
{
    // Decrypt the cypher text generated using the key and display
    cout << "After decryption : ";
    for (int i = 0; i < cText.length(); i++)
    {
        cout << deKeyMap[cText[i]];
    }
    cout << endl;
}
void ShowFrequency(string pText, unordered_map<char, char> enKeyMap)
{


    float fTable[26] = {0.000};
    int pTextLen = pText.length();
    for (int i = 0; i < pTextLen; i++)
    {
        int idx = pText[i] - 'a';
        fTable[idx]++;
    }
```

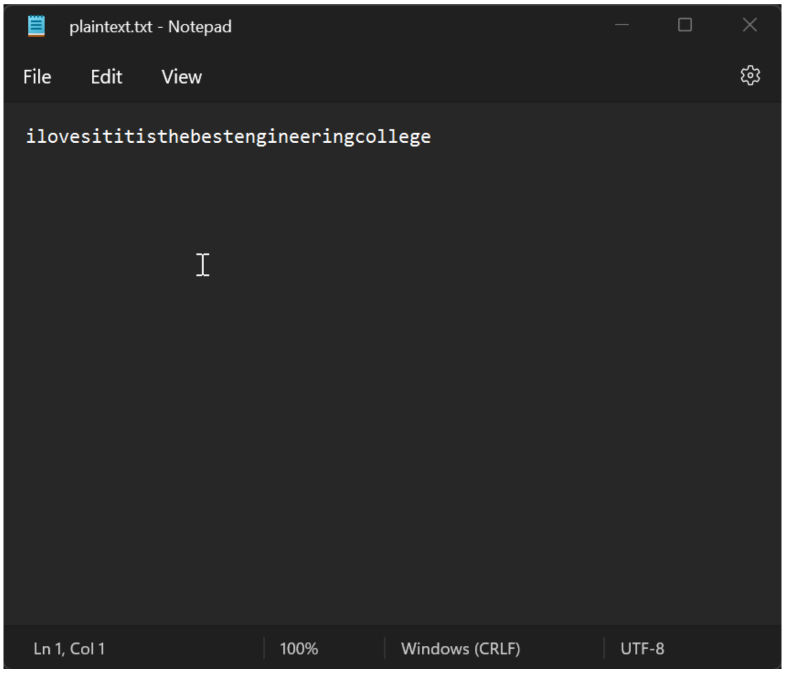
```
cout << endl << "Frequency \t Plain Char \t CipherChar" << endl;


for (int i = 0; i < 26; i++)
    {
        char c = (char)('a' + i);
        cout << fixed << setprecision(3) << (fTable[i] / pTextLen) << "\t\t   " << c << "\t\t   " << enKeyMap[c] << endl;
    }
}
int main()
{

    string key = "";
    string plainText = "";
    string cipherText = "";
    ifstream fin("plaintext.txt");
    ofstream fout("ciphertext.txt");
    unordered_map<char, char> enKeyMap, deKeyMap;

    key = GenKey();

    MapKey(enKeyMap, deKeyMap, key);

    fin >> plainText;

    Encrypt(plainText, cipherText, enKeyMap);

    cout << "CipherText :" << cipherText << endl;
    fout << cipherText;
    Decrypt(cipherText, deKeyMap);

    ShowFrequency(plainText, enKeyMap);
    return 0;
}
```

OUTPUT:



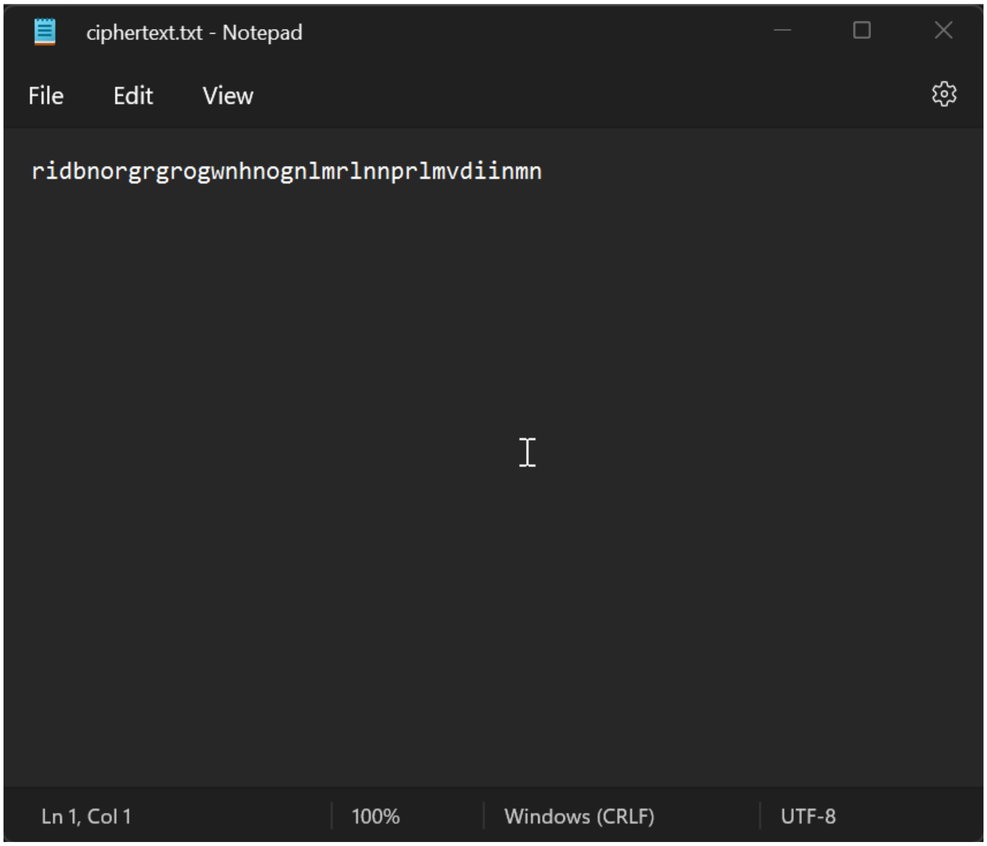
```
anudeep@Anudeep:/mnt/d/c,c++$ g++ monoalphabetic-cipher.cpp
anudeep@Anudeep:/mnt/d/c,c++$ ./a.out
Key :shvtnzmwrauijldxypogkbefcq
CipherText :ridbnorgrgrogwnhnognlmrlnnprlmvdiinmn
After decryption : ilovesititisthebestengineeringcollege

Frequency        Plain Char      CipherChar
0.000              a               s
0.027              b               h
0.027              c               v
0.000              d               t
0.216              e               n
0.000              f               z
0.081              g               m
0.027              h               w
0.162              i               r
0.000              j               a
0.000              k               u
0.081              l               i
0.000              m               j
0.081              n               l
0.054              o               d
0.000              p               x
0.000              q               y
0.027              r               p
0.081              s               o
0.108              t               g
0.000              u               k
0.027              v               b
0.000              w               e
0.000              x               f
0.000              y               c
```

Plaintext.txt:



```
ilovesititisthebestengineeringcollege
```

Ciphertext.txt



```
ridbnorgrgrogwnhnognlmrlnnprlmvdiinmn
```

SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMKUR-572103

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CRYPTOGRAPHY AND NETWORK SECURITY LAB (7RCSL01)

Student Name: ANVS Anudeep	**USN: 1SI19CS017**	**Batch No: A1**	**Date: 21/11/2022**

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.	Dr AS Poornima	
2.	Ravi V	

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 cipher text pair

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

Playfair Cipher:

Construct **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 alphabetic order. The letters I and J count as one letter.

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

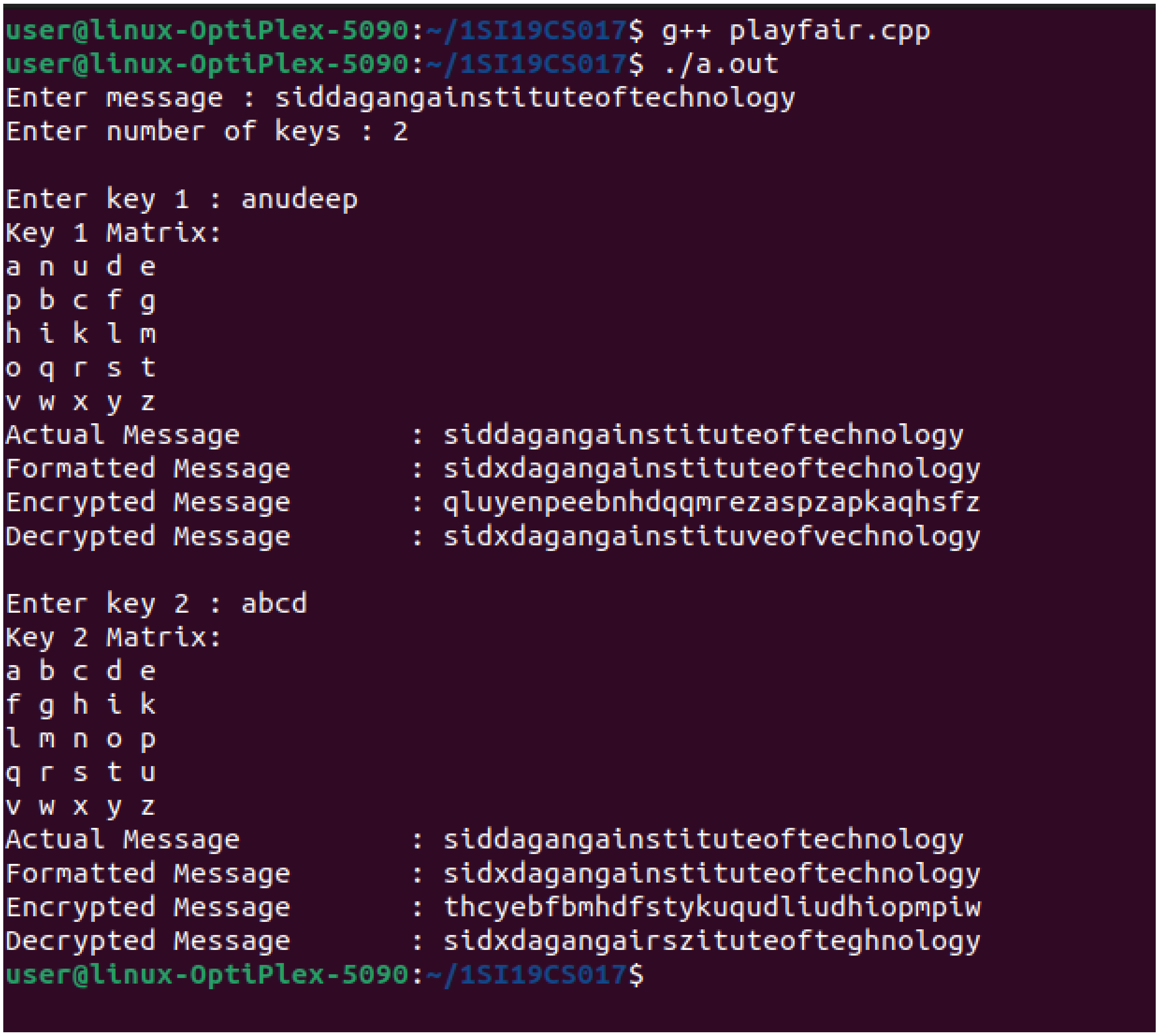
1. Repeating plaintext letters that are in the same pair are separated with a filler letter, such as x, so that balloon would be treated as ba lx lo on.
2. Two plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row circularly 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.

```
#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]);
            ctext += mat[x1][(y1 + 1) % 5];
            ctext += mat[x2][(y2+1)%5];
        }
        else if( y1 == y2 )
        {
            ctext += mat[ (x1+1)%5 ][ y1 ];
            ctext += mat[x2][(y2+1)%5];
            // ctext.append(1, mat[ (x1+1)%5 ][ y1 ]);
            // ctext.append(1, mat[ (x2+1)%5 ][ y2 ]);
        }
```

```
                else
                {
                        ctext += mat[x1][y2];
                        ctext += mat[x2][y1];
                        // 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; // number of keys
```

```
    cout << "Enter number of keys : "; cin >> n;

    string key[n];
    for(int i=0; i<n; i++)
    {
        cout<< "\nEnter key " << i+1 << " : " << key[i];
        cin >> key[i];

        generateMatrix(key[i]);

        cout << "Key " << 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 \t\t: " << plaintext << endl;

        string fmsg = formatMessage(plaintext);
        cout << "Formatted Message \t: " << fmsg << endl;

        string ciphertext = encrypt(fmsg);
        cout << "Encrypted Message \t: " << ciphertext << endl;




        string decryptmsg = Decrypt(ciphertext);
        cout<< "Decrypted Message \t: " << decryptmsg << endl;
    }
}
```

OUTPUT:



```
user@linux-OptiPlex-5090:~/1SI19CS017$ g++ playfair.cpp
user@linux-OptiPlex-5090:~/1SI19CS017$ ./a.out
Enter message : siddagangainstituteoftechnology
Enter number of keys : 2

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

Enter key 2 : abcd
Key 2 Matrix:
a b c d e
f g h i k
l m n o p
q r s t u
v w x y z
Actual Message          : siddagangainstituteoftechnology
Formatted Message       : sidxdagangainstituteoftechnology
Encrypted Message       : thcyebfbmhdfstykuqudliudhiopmpiw
Decrypted Message       : sidxdagangairszituteofteghnology
user@linux-OptiPlex-5090:~/1SI19CS017$
```

SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMKUR-572103

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CRYPTOGRAPHY AND NETWORK SECURITY LAB (7RCSL01)

Student Name: ANVS Anudeep	**USN: 1SI19CS017**	**Batch No: A1**	**Date: 05/12/2022**

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.	**Dr AS Poornima**	
2.	**Ravi V**	

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

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

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

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

C = E(K, P) = 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.

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

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

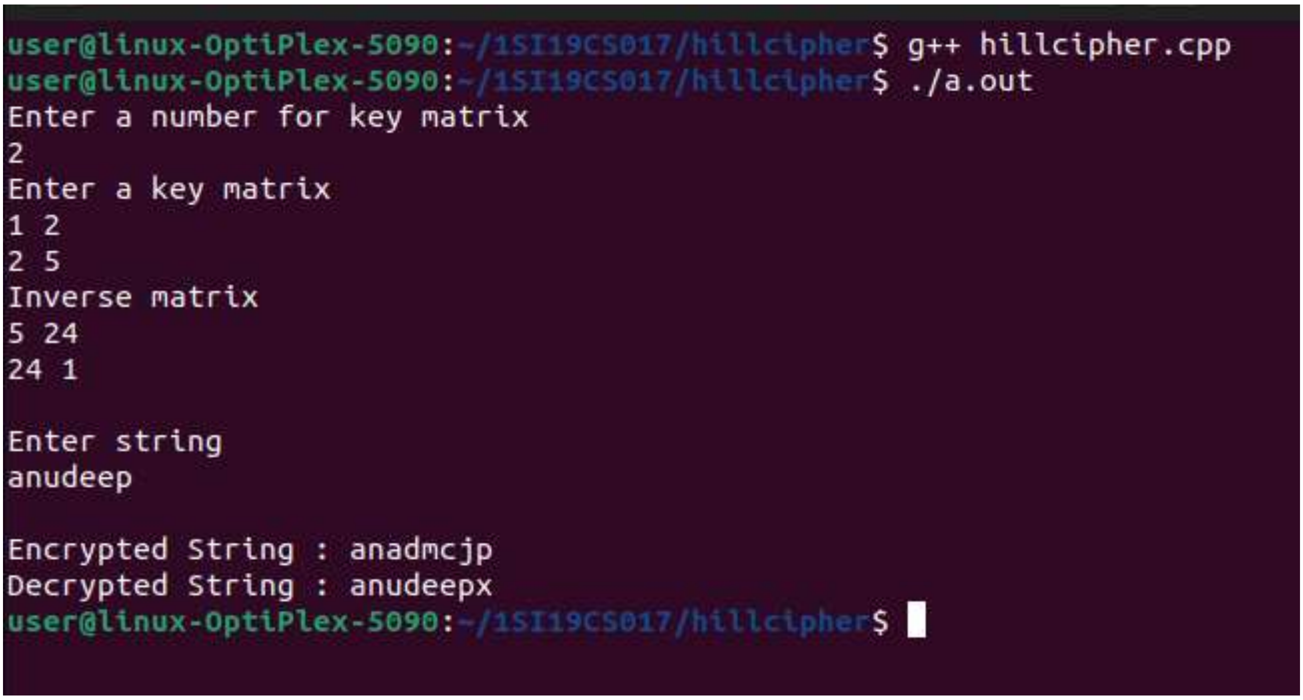
If a square matrix A has a nonzero determinant, then the inverse of the matrix is computed as $[A^{-1}]_{ij} = (\det A)^{-1}(-1)^{i+j}(D_{ji})$, where (D_{ji}) is the sub determinant formed by deleting the 'j'th row and the' i'th column of A, det(A) is the determinant of A, and $(\det A)^{-1}$ is the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

```
#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]) ;
}
```

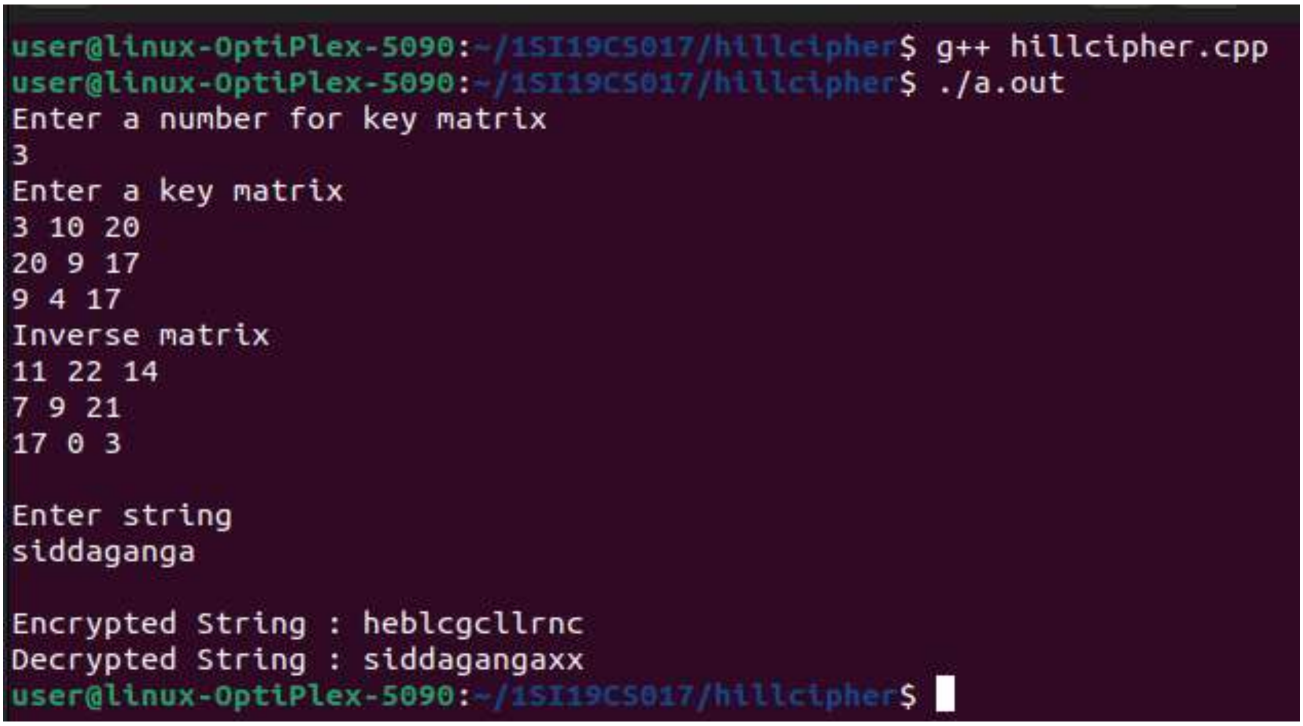
```
void multiplyMatrices(int a[1000][3] , int a_rows , int a_cols , int b[1000][3] , int b_rows , int b_cols , int res[1000][3])
{
    for(int i=0 ; i < a_rows ; i++)
    {
        for(int j=0 ; j < b_cols ; j++)
        {
            for(int k=0 ; k < b_rows ; k++)
            {
                res[i][j] += 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};
    //p = c.k-1.mod26

    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: " ;
	cin >> pt;

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

	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;
	return 0;
}
```



```
user@linux-OptiPlex-5090:~/1SI19CS017/hillcipher$ g++ hillcipher.cpp
user@linux-OptiPlex-5090:~/1SI19CS017/hillcipher$ ./a.out
Enter a number for key matrix
2
Enter a key matrix
1 2
2 5
Inverse matrix
5 24
24 1

Enter string
anudeep

Encrypted String : anadmcjp
Decrypted String : anudeepx
user@linux-OptiPlex-5090:~/1SI19CS017/hillcipher$ 
```



```
user@linux-OptiPlex-5090:~/1SI19CS017/hillcipher$ g++ hillcipher.cpp
user@linux-OptiPlex-5090:~/1SI19CS017/hillcipher$ ./a.out
Enter a number for key matrix
3
Enter a key matrix
3 10 20
20 9 17
9 4 17
Inverse matrix
11 22 14
7 9 21
17 0 3

Enter string
siddaganga

Encrypted String : heblcgcllrnc
Decrypted String : siddagangaxx
user@linux-OptiPlex-5090:~/1SI19CS017/hillcipher$ 
```

SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMKUR-572103

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CRYPTOGRAPHY AND NETWORK SECURITY LAB (7RCSL01)

Student Name: ANVS Anudeep	**USN: 1SI19CS017**	**Batch No: A1**	**Date: 28/11/22**

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.	**Dr AS Poornima**	
2.	**Ravi V**	

Question No: 4

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

Transposition technique:

Encryption:

- Construct a matrix/rectangle in which column is represented by key.
- Message/ plain text must be filled row by row over a specified number of columns.
- Read the content of the matrix column by column in order of the given key to get the cipher text.

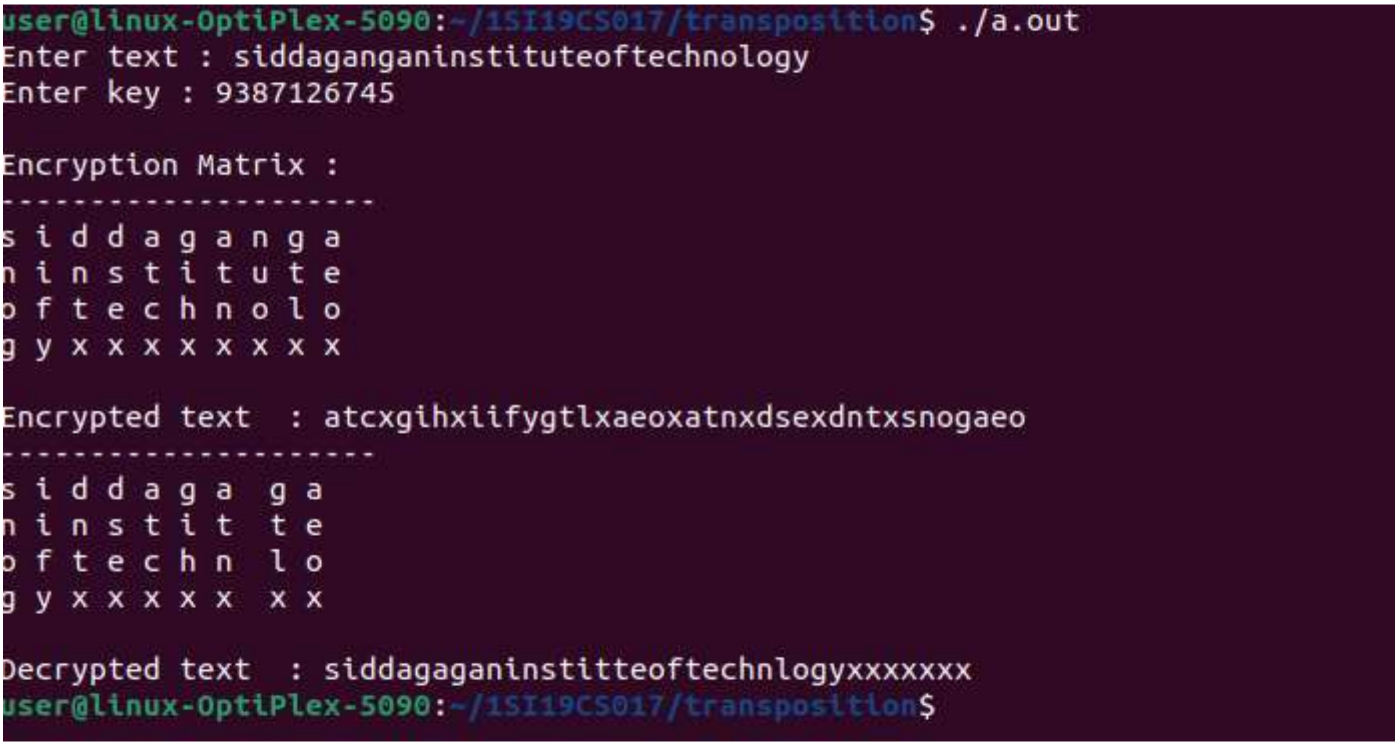
Decryption :

- Find the number of rows and colunms in a matrix :

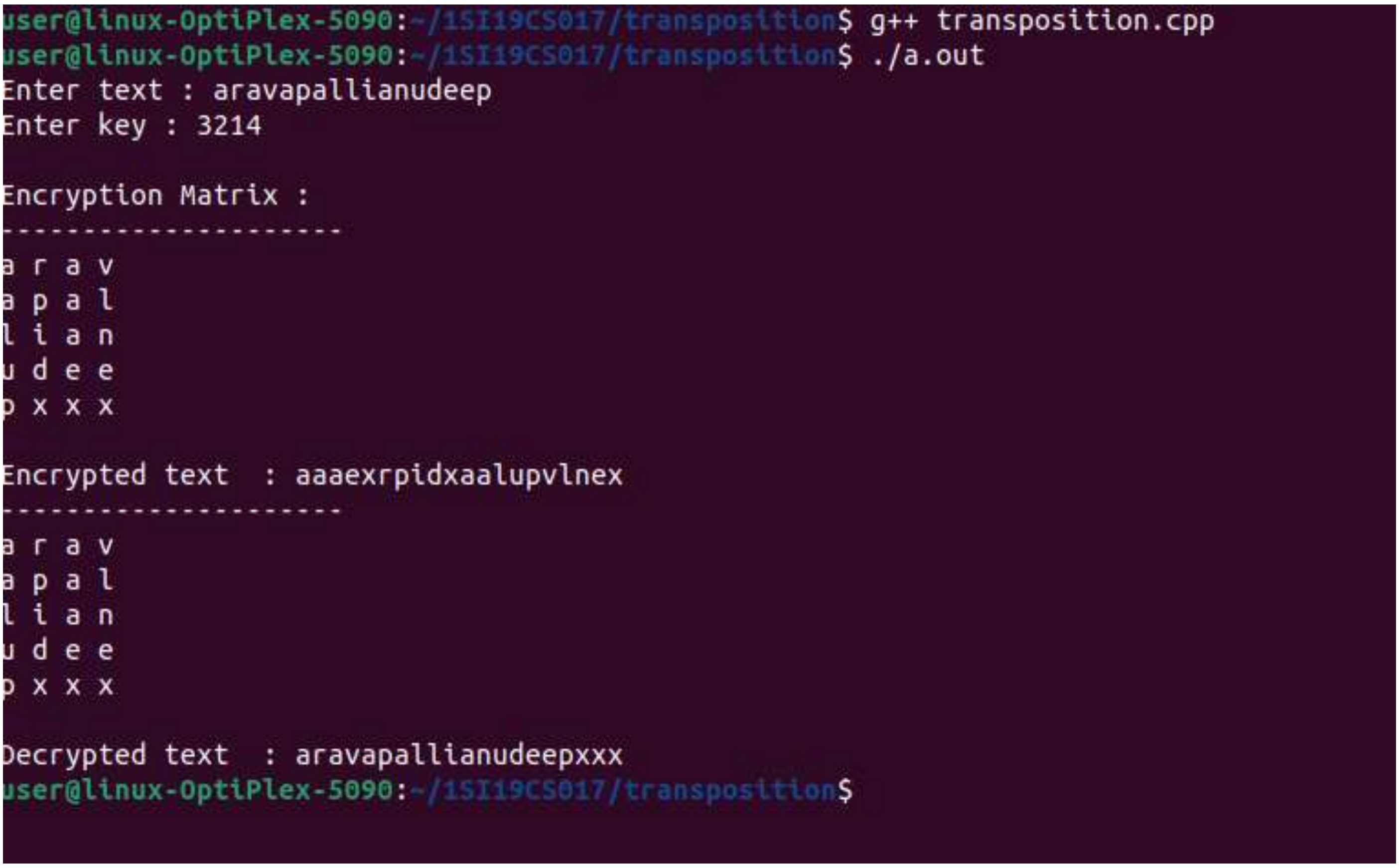
  No of Rows= Length of the Cipher/ Length of the Key.

  No of Columns is represented by digits in the key.
- Fill the cipher text in a matrix column by column orderly.
- Read the content of the matrix column by column in order to get the plain text.

```
#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<<"\nEncrytion 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<<"\nDecrypted 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: ";
        cin>>plaintext;
        cout<<"enter key: ";
        cin>>key;
        ciphertext=encrypt(plaintext,key);
        cout<<"\nencrypted text\t: "<<ciphertext<<endl;

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



```
user@linux-OptiPlex-5090:~/1SI19CS017/transposition$ ./a.out
Enter text : siddaganganinstituteoftechnology
Enter key : 9387126745

Encryption Matrix :
--------------------
s i d d a g a n g a
n i n s t i t u t e
o f t e c h n o l o
g y x x x x x x x x

Encrypted text  : atcxgihxiifygtlxaeoxatnxdsexdntxsnogaeo
--------------------
s i d d a g a   g a
n i n s t i t   t e
o f t e c h n   l o
g y x x x x x   x x

Decrypted text  : siddagaganinstitteoftechnlogyxxxxxxx
user@linux-OptiPlex-5090:~/1SI19CS017/transposition$
```



```
user@linux-OptiPlex-5090:~/1SI19CS017/transposition$ g++ transposition.cpp
user@linux-OptiPlex-5090:~/1SI19CS017/transposition$ ./a.out
Enter text : aravapallianudeep
Enter key : 3214

Encryption Matrix :
--------------------
a r a v
a p a l
l i a n
u d e e
p x x x

Encrypted text  : aaaexrpidxaalupvlnex
--------------------
a r a v
a p a l
l i a n
u d e e
p x x x

Decrypted text  : aravapallianudeepxxx
user@linux-OptiPlex-5090:~/1SI19CS017/transposition$
```

SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMKUR-572103

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CRYPTOGRAPHY AND NETWORK SECURITY LAB (7RCSL01)

Student Name: ANVS Anudeep	**USN: 1SI19CS017**	**Batch No: A1**	**Date: 19/12/2022**

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.	Dr AS Poornima	
2.	Ravi V	

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.

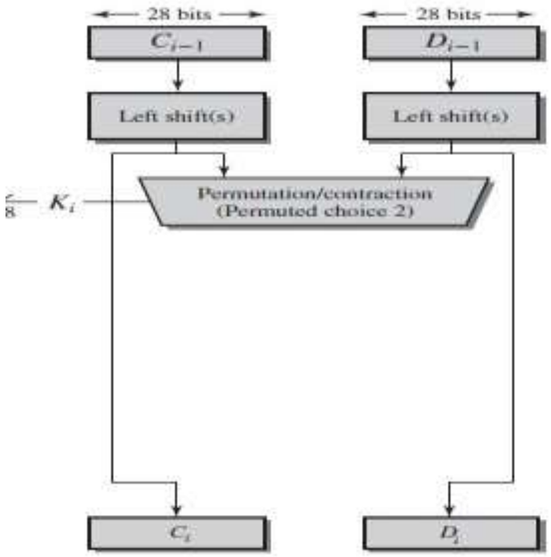
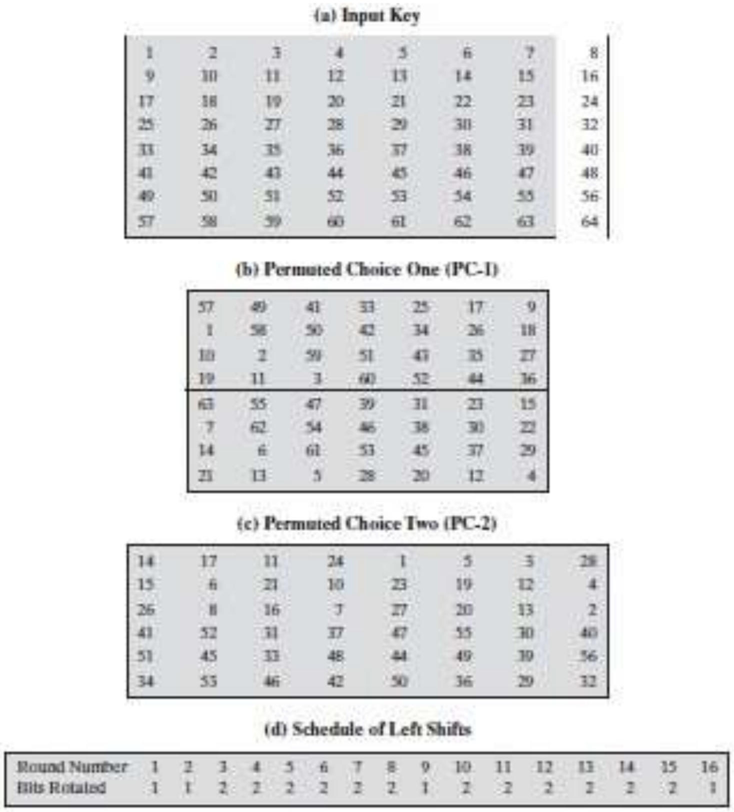


Figure: DES key Schedule Calculation



(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

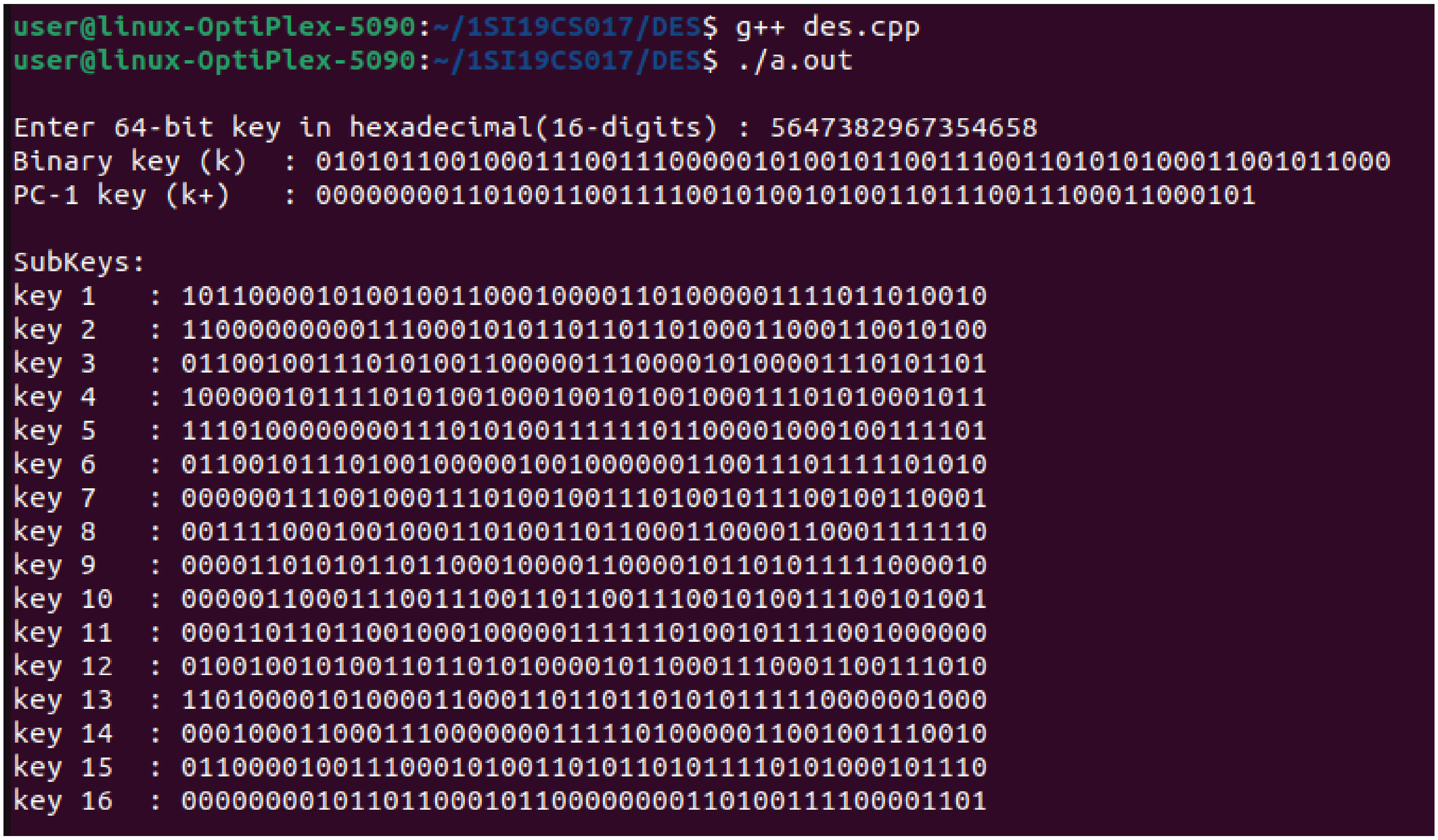
(d) Schedule of Left Shifts

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

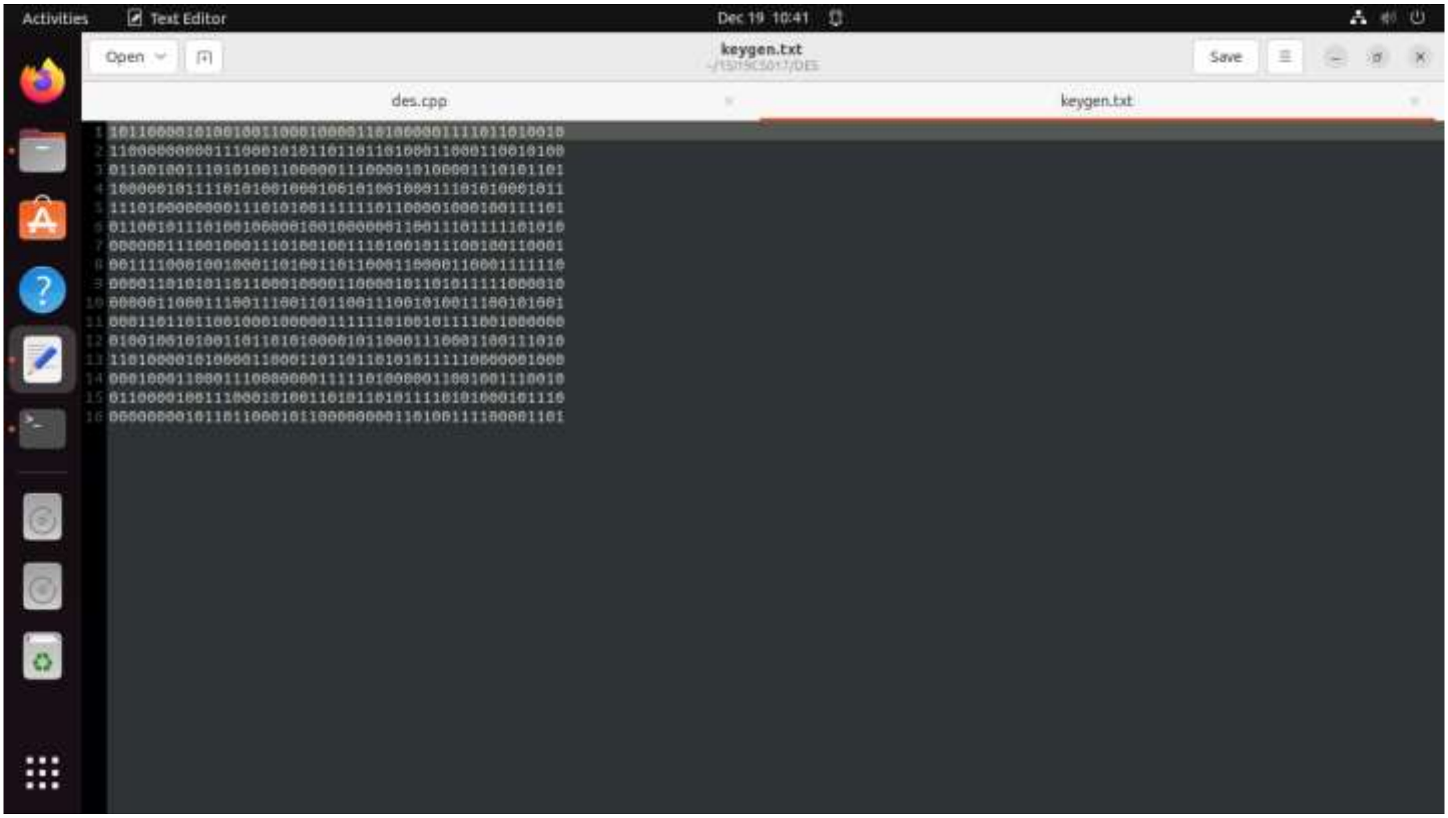
Tables: DES key Schedule Calculation

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



```
user@linux-OptiPlex-5090:~/1SI19CS017/DES$ g++ des.cpp
user@linux-OptiPlex-5090:~/1SI19CS017/DES$ ./a.out

Enter 64-bit key in hexadecimal(16-digits) : 5647382967354658
Binary key (k)  : 0101011001000111001110000010100101100111001101010100011001011000
PC-1 key (k+)   : 00000000110100110011110010100101001101110011100011000101

SubKeys:
key 1   : 101100000101001001100010000110100000111101101010010
key 2   : 110000000000111000101011011011010001100011001010 0
key 3   : 011001001110101001100000111000010100001110101101
key 4   : 100000101111010100100010010100100011101010001011
key 5   : 111010000000011101010011111101100001000100111101
key 6   : 011001011101001000001001000000110011101111101010
key 7   : 000000111001000111010010011101001011100100110001
key 8   : 001111000100100011010011011000110000110001111110
key 9   : 000011010101101100010000110000101101011111000010
key 10  : 000001100011100111001101100111001010011100101001
key 11  : 000110110110010001000001111110100101111001000000
key 12  : 010010010100110110101000010110001110001100111010
key 13  : 110100001010000110001101101101010111110000001000
key 14  : 000100011000111000000001111101000001100100111001 0
key 15  : 011000010011100010100110101101011110101000101110
key 16  : 000000001011011000101100000000110100111100001101
```



```
101100000101001001100010000110100000111101101010010
110000000000111000101011011011010001100011001010 0
011001001110101001100000111000010100001110101101
100000101111010100100010010100100011101010001011
111010000000011101010011111101100001000100111101
011001011101001000001001000000110011101111101010
000000111001000111010010011101001011100100110001
001111000100100011010011011000110000110001111110
000011010101101100010000110000101101011111000010
000001100011100111001101100111001010011100101001
000110110110010001000001111110100101111001000000
010010010100110110101000010110001110001100111010
110100001010000110001101101101010111110000001000
000100011000111000000001111101000001100100111001 0
011000010011100010100110101101011110101000101110
000000001011011000101100000000110100111100001101
```

SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMKUR-572103

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CRYPTOGRAPHY AND NETWORK SECURITY LAB (7RCSL01)

Student Name: ANVS Anudeep	**USN: 1SI19CS017**	**Batch No: A1**	**Date:26/12/2022**

Evaluation:

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

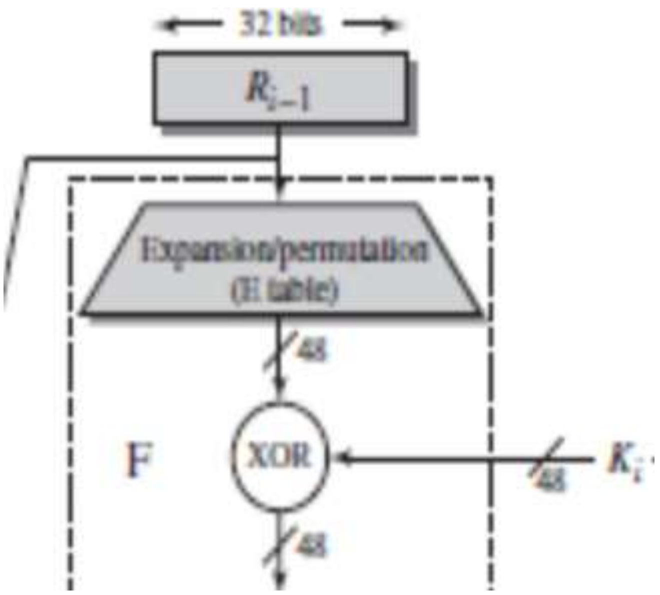
Sl.No	Name of the Faculty In-Charge	Signature
1.	**Dr AS Poornima**	
2.	**Ravi V**	

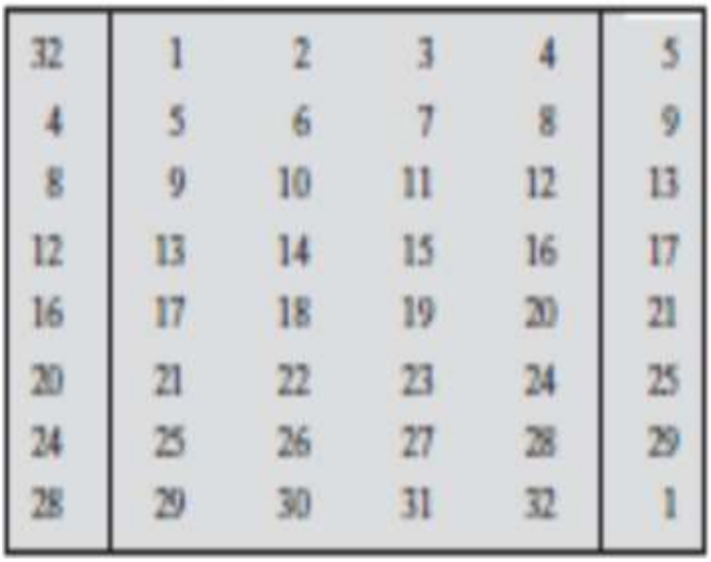
Question No: 6

i) Given 64-bit output of (i-1)th round of DES, 48-bit ith round key K_i and E table, find the 48-bit input for S-box.

ii) Given 48-bit input to S-box and permutation table P, find the 32-bit output R_i of ith round of DES algorithm.

i) **Algorithm:** Follow the flow-chart and tables given below.

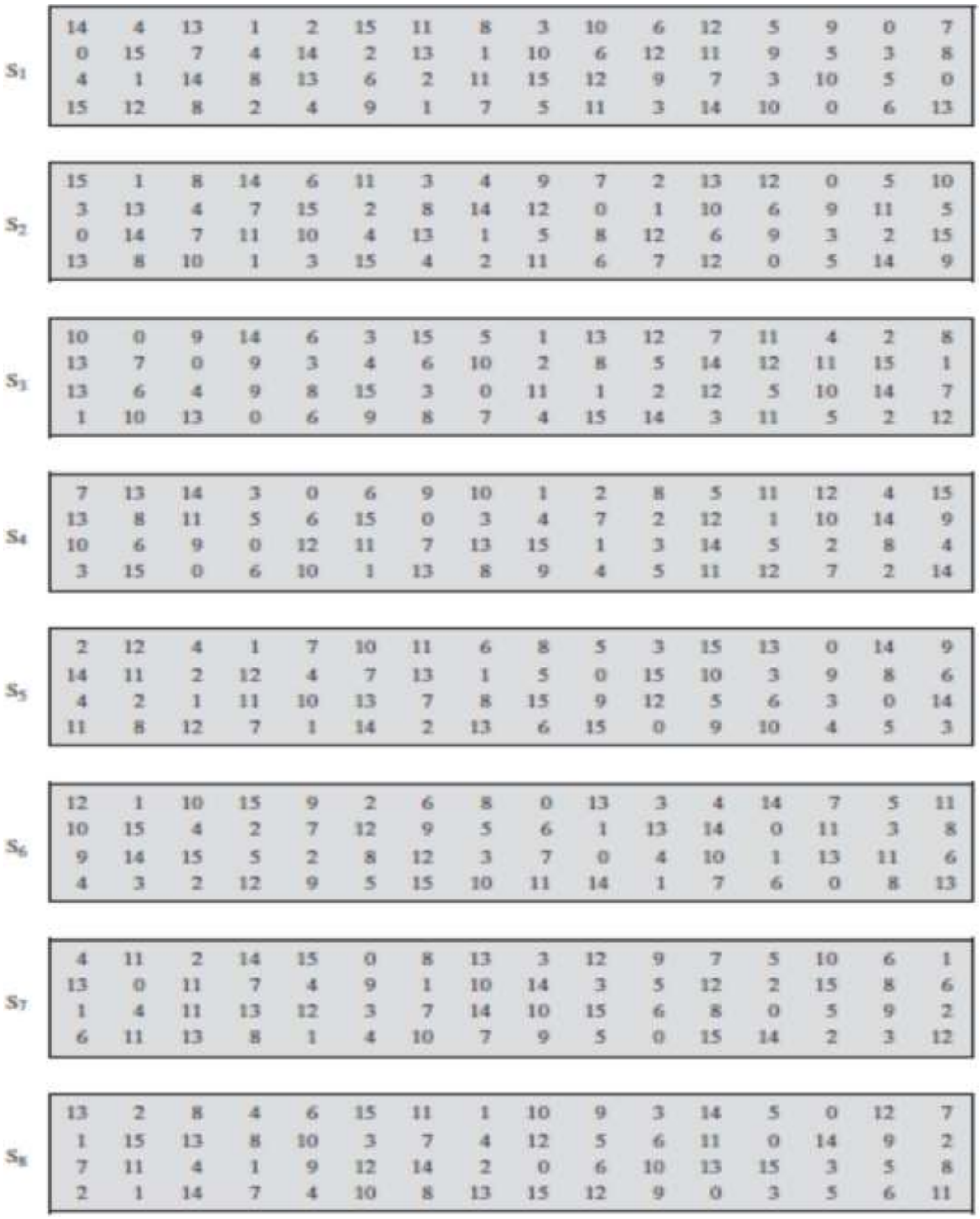




32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Figure: Generation of 48-bit input for S-box.

Table: Expansion Permutation



S_1	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

S_2	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

S_3	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

S_4	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

S_5	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

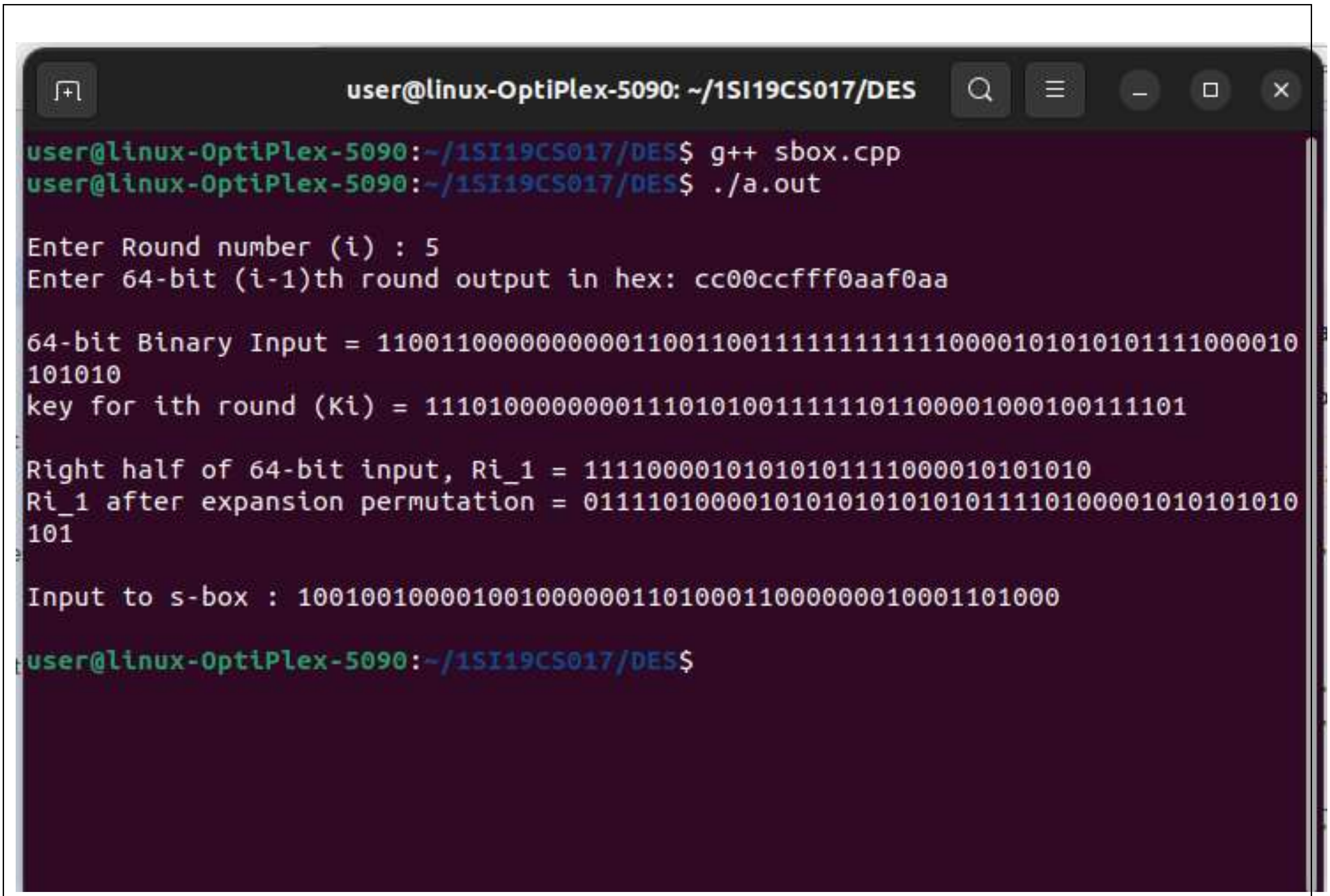
S_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7	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

S_8	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

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

int expPermute[] = {
                        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 };
string expansionPermute(string input)
{

        string res = "";
        for(int i=0; i<48; i++)
        {
                res += input[expPermute[i]-1];
        }
        return res;
}
string XOR(string input1, string input2)
{
        string res = "";
        for(int i=0; i<input1.length(); i++)
        {
                res += (input1[i] == input2[i]) ? "0" : "1";
        }
        return res;
}
int main()
{
        int i; // round i
        unsigned long long hexInput;
        string Ki; // ith round key
        ifstream fin;

        cout << "\nEnter Round number (i) : ";
        cin >> i;

        cout << "Enter 64-bit (i-1)th round output in hex: " ;
        cin >> hex >> hexInput;

        string input = bitset<64>(hexInput).to_string();
        fin.open("keygen.txt");
```

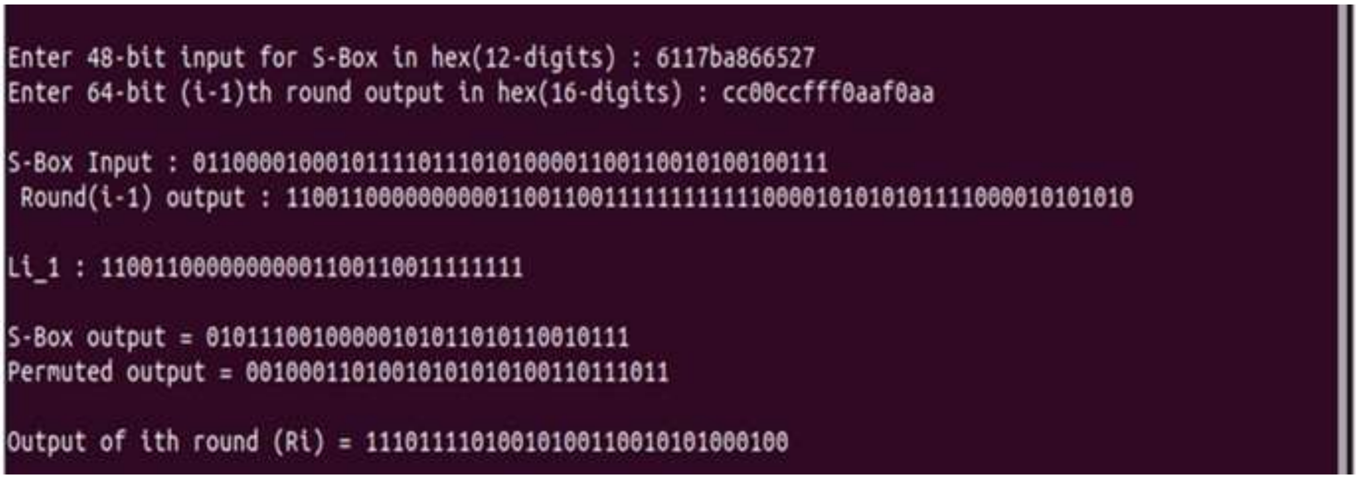
```
	for(int j=1; j<=i; j++)
	{
		fin >> Ki;
	}

	if(Ki.length() == 0)
	{
		cout << "\nkeygen.txt not found !!! \n" << endl;
		exit(1);
	}
	cout << "\n64-bit Binary Input = " << input << endl ;
	cout << "key for ith round (Ki) = " << Ki << endl ;

	string Ri_1 = input.substr(32,32); // 32 bit Right half of input R[i-1]
	cout << "\nRight half of 64-bit input, Ri_1 = " << Ri_1 << endl;

	string R48 = expansionPermute(Ri_1);
	cout << "Ri_1 after expansion permutation = " << R48 << endl;

	string sBoxInput = XOR(R48, Ki);
	cout << "\nInput to s-box : " << sBoxInput << endl << endl;
}
```



```
user@linux-OptiPlex-5090:~/1SI19CS017/DES$ g++ sbox.cpp
user@linux-OptiPlex-5090:~/1SI19CS017/DES$ ./a.out

Enter Round number (i) : 5
Enter 64-bit (i-1)th round output in hex: cc00ccfff0aaf0aa

64-bit Binary Input = 1100110000000000110011001111111111110000101010101111000010
101010
key for ith round (Ki) = 111010000000011101010011111101100001000100111101

Right half of 64-bit input, Ri_1 = 11110000101010101111000010101010
Ri_1 after expansion permutation = 011110100001010101010101011110100001010101010
101

Input to s-box : 100100100001001000000110100011000000010001101000

user@linux-OptiPlex-5090:~/1SI19CS017/DES$
```

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

unsigned int sBoxes[8][64] = {
	{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}
};

int permTable[] = {
	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 };
```

```
string substitution(string input)
{
    string res = "";
    for(int i=0; i<8; i++)
    {
        string sInput = input.substr(6*i, 6) ;
        int row = bitset<2>( sInput.substr(0,1) + sInput.substr(5,1) ).to_ulong() ;
        int col = bitset<4>( sInput.substr(1,4) ).to_ulong() ;
        res += bitset<4>(sBoxes[i][row*16+col]).to_string() ;
    }
    return res;
}

string permute(string input)
{
    string res = "";
    for(int i=0; i<32; i++)
    {
        res += input[permTable[i]-1];
    }
    return res;
}

string XOR(string input1, string input2)
{
    string res = "";
    for(int i=0; i<input1.length(); i++)
    {
        res += (input1[i] == input2[i]) ? "0" : "1";
    }
    return res;
}

int main()
{
    unsigned long long hexSBoxInput, hexInput;
    cout << "\nEnter 48-bit input for S-Box in hex(12-digits) : " ;
    cin >> hex >> hexSBoxInput;
    cout << "Enter 64-bit (i-1)th round output in hex(16-digits) : " ;
    cin >> hex >> hexInput;
    string sBoxinput = bitset<48>(hexSBoxInput).to_string();
    cout << "\nS-Box Input : " << sBoxinput << endl;
    string input = bitset<64>(hexInput).to_string();
    cout << " Round(i-1) output : " << input << endl;
    string Li_1 = input.substr(0,32);
    cout << "\nLi_1 : " << Li_1 << endl;
    string sBoxOutput = substitution(sBoxinput);
    cout << "\nS-Box output = " << sBoxOutput << endl;
    string P = permute(sBoxOutput);
    cout << "Permuted output = " << P << endl;
    string Ri = XOR(P, Li_1);
    cout << "\nOutput of ith round (Ri) = " << Ri << endl << endl;
}
```



```
Enter 48-bit input for S-Box in hex(12-digits) : 6117ba866527
Enter 64-bit (i-1)th round output in hex(16-digits) : cc00ccfff0aaf0aa

S-Box Input : 011000010001011110111010100001100110010100100111
 Round(i-1) output : 1100110000000000110011001111111111110000101010101111000010101010

Li_1 : 11001100000000001100110011111111

S-Box output = 01011100100000101011010110010111
Permuted output = 00100011010010101010100110111011

Output of ith round (Ri) = 11101111010010100110010101000100
```

SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMKUR-572103
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CRYPTOGRAPHY AND NETWORK SECURITY LAB (7RCSL01)

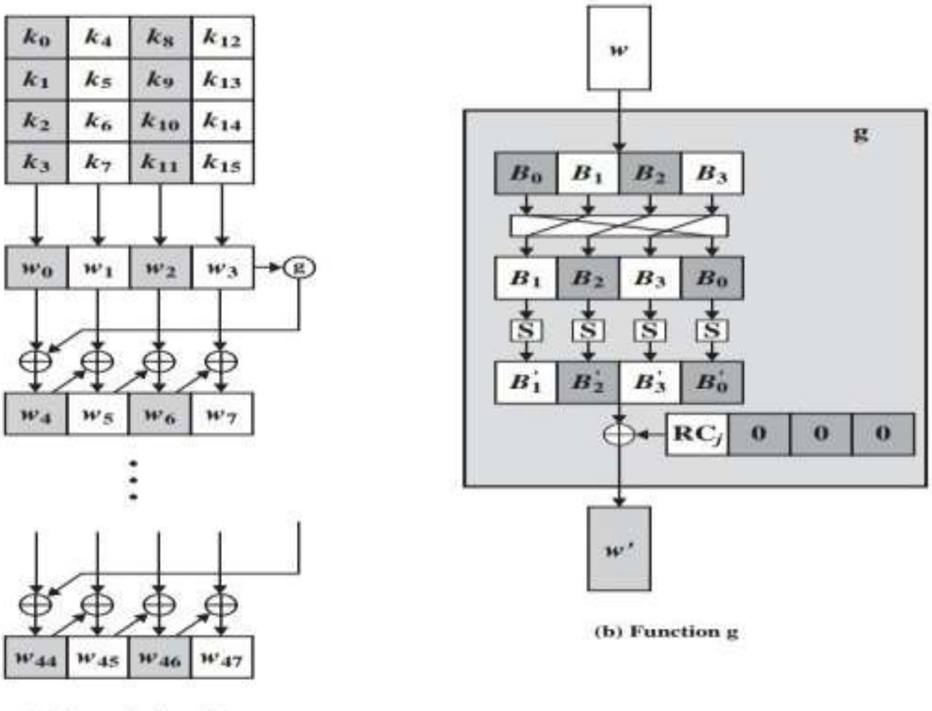
Student Name: ANVS Anudeep	**USN: 1SI19CS017**	**Batch No: A1**	**Date: 02-01-2023**
Evaluation:			

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.	Dr AS Poornima	
2.	Ravi V	

Question No: 7. Consider the 128 bits initial key and expand it to 10 different keys each of size 128 bits using AES key expansion technique.

Algorithm:



k_0 k_4 k_8 k_{12}
k_1 k_5 k_9 k_{13}
k_2 k_6 k_{10} k_{14}
k_3 k_7 k_{11} k_{15}
w_0 w_1 w_2 w_3
g
w_4 w_5 w_6 w_7
w_{44} w_{45} w_{46} w_{47}
w
g
B_0 B_1 B_2 B_3
B_1 B_2 B_3 B_0
S S S S
B'_1 B'_2 B'_3 B'_0
RC_j 0 0 0
w'

(b) Function g

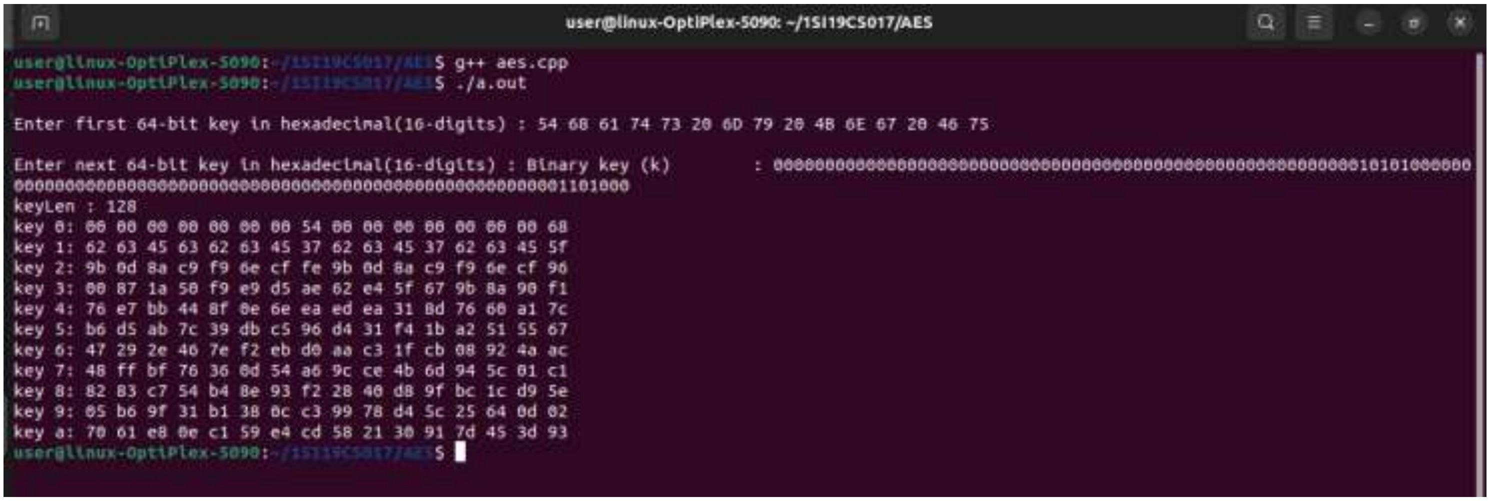
(a) Overall algorithm

Figure 1.1 AES Key Expansion

CODE:-

```
#include <bits/stdc++.h>
using namespace std;
unsigned long long sbox[16][16] = {
    {0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76},
    {0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0},
    {0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15},
    {0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75},
    {0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84},
    {0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf},
    {0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8},
    {0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2},
    {0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73},
    {0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb},
    {0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79},
    {0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08},
    {0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a},
    {0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e},
    {0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf},
    {0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16}
};
unsigned long long Rcon[10] = {
    0x01000000, 0x02000000, 0x04000000, 0x08000000, 0x10000000, 0x20000000, 0x40000000, 0x80000000,
0x1b000000, 0x36000000
};
string w[44];

string rotLeft(string word)
{
    return word.substr(8) + word.substr(0,8);
}
string SBoxFun(string word)
{
    string res = "";
    for(int i=0; i<4; i++){
        string byte = word.substr(i*8, 8);
        int row = bitset<4>( byte.substr(0,4) ).to_ulong();
        int col = bitset<4>( byte.substr(4,4) ).to_ulong();
        res += bitset<8>(sbox[row][col]).to_string();
    }
    return res;
}
string XOR(string x, string y){
    string res = "";
    for(int i=0; i<x.length(); i++)
    {
        res += (x[i] == y[i]) ? "0" : "1";
    }
    return res;
```

```
  }
  int main()
  {
        unsigned long long hexkey1, hexkey2;
        cout << "\nEnter first 64-bit key in hexadecimal(16-digits) : " ;
        cin >> hex >> hexkey1;
        cout << "\nEnter next 64-bit key in hexadecimal(16-digits) : " ;
        cin >> hex >> hexkey2;
        string key = bitset<64>(hexkey1).to_string() + bitset<64>(hexkey2).to_string();
        cout << "Binary key (k) \t: " << key << endl;
        cout << "keyLen : " << key.length() << endl;
        for(int i=0; i<4; i++){
                w[i] = key.substr(i*32,32);
        }
        for(int i=4; i<44; i++)
        {
                string first = w[i-4];
                string second = w[i-1];
                if(i % 4 == 0)
                {
                        second = rotLeft(second);
                        second = SBoxFun(second);
                        string tmp = bitset<32>(Rcon[i/4]).to_string();
                        second = XOR(second, tmp);
                }
                w[i] = XOR(first, second);
        }
        string keys[11] = {""};
        for(int i=0; i<44; i++)
        {
                keys[i/4] += w[i];
        }
        for(int i=0; i<11; i++)
        {
                cout << "Key " << i << ": ";
                for(int j=0; j<16;j++)
                {
                        cout << setfill('0') << setw(2)<<hex<<
                        bitset<8>(keys[i].substr(j*8,8)).to_ulong() <<" ";
                }
                cout <<endl;
        }
        return 0;
}
```



```
user@linux-OptiPlex-5090:~/1SI19CS017/AES$ g++ aes.cpp
user@linux-OptiPlex-5090:~/1SI19CS017/AES$ ./a.out

Enter first 64-bit key in hexadecimal(16-digits) : 54 68 61 74 73 20 6D 79 20 4B 6E 67 20 46 75

Enter next 64-bit key in hexadecimal(16-digits) : Binary key (k)        : 00000000000000000000000000000000000000000000000000000000010101000000
000000000000000000000000000000000000000000000000000001101000
keyLen : 128
key 0: 00 00 00 00 00 00 00 54 00 00 00 00 00 00 00 68
key 1: 62 63 45 63 62 63 45 37 62 63 45 37 62 63 45 5f
key 2: 9b 0d 8a c9 f9 6e cf fe 9b 0d 8a c9 f9 6e cf 96
key 3: 00 87 1a 50 f9 e9 d5 ae 62 e4 5f 67 9b 8a 90 f1
key 4: 76 e7 bb 44 8f 0e 6e ea ed ea 31 8d 76 60 a1 7c
key 5: b6 d5 ab 7c 39 db c5 96 d4 31 f4 1b a2 51 55 67
key 6: 47 29 2e 46 7e f2 eb d0 aa c3 1f cb 08 92 4a ac
key 7: 48 ff bf 76 36 0d 54 a6 9c ce 4b 6d 94 5c 01 c1
key 8: 82 83 c7 54 b4 8e 93 f2 28 40 d8 9f bc 1c d9 5e
key 9: 05 b6 9f 31 b1 38 0c c3 99 78 d4 5c 25 64 0d 02
key a: 70 61 e8 0e c1 59 e4 cd 58 21 30 91 7d 45 3d 93
user@linux-OptiPlex-5090:~/1SI19CS017/AES$ 
```

SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMKUR-572103

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CRYPTOGRAPHY AND NETWORK SECURITY LAB (7RCSL01)

Student Name: ANVS Anudeep	**USN: 1SI19CS017**	**Batch No:A1**	**Date:02-01-2023**
Evaluation:			

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

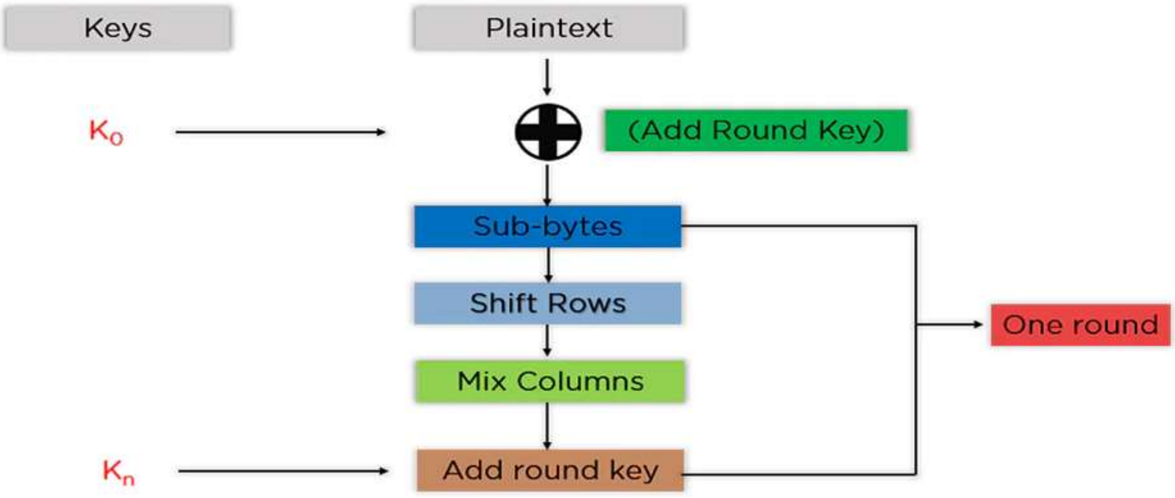
Sl.No	**Name of the Faculty In-Charge**	**Signature**
1.	Dr AS Poornima	
2.	Ravi V	

Question No: 8. Consider a message of 16 bytes (128 bits) and perform XOR operation with an initial round key [W0, W1, W2, W3] of size 128 bits to generate a state array in AES. W.r.t generated state array of size 128 bits, perform the following operations in each round.

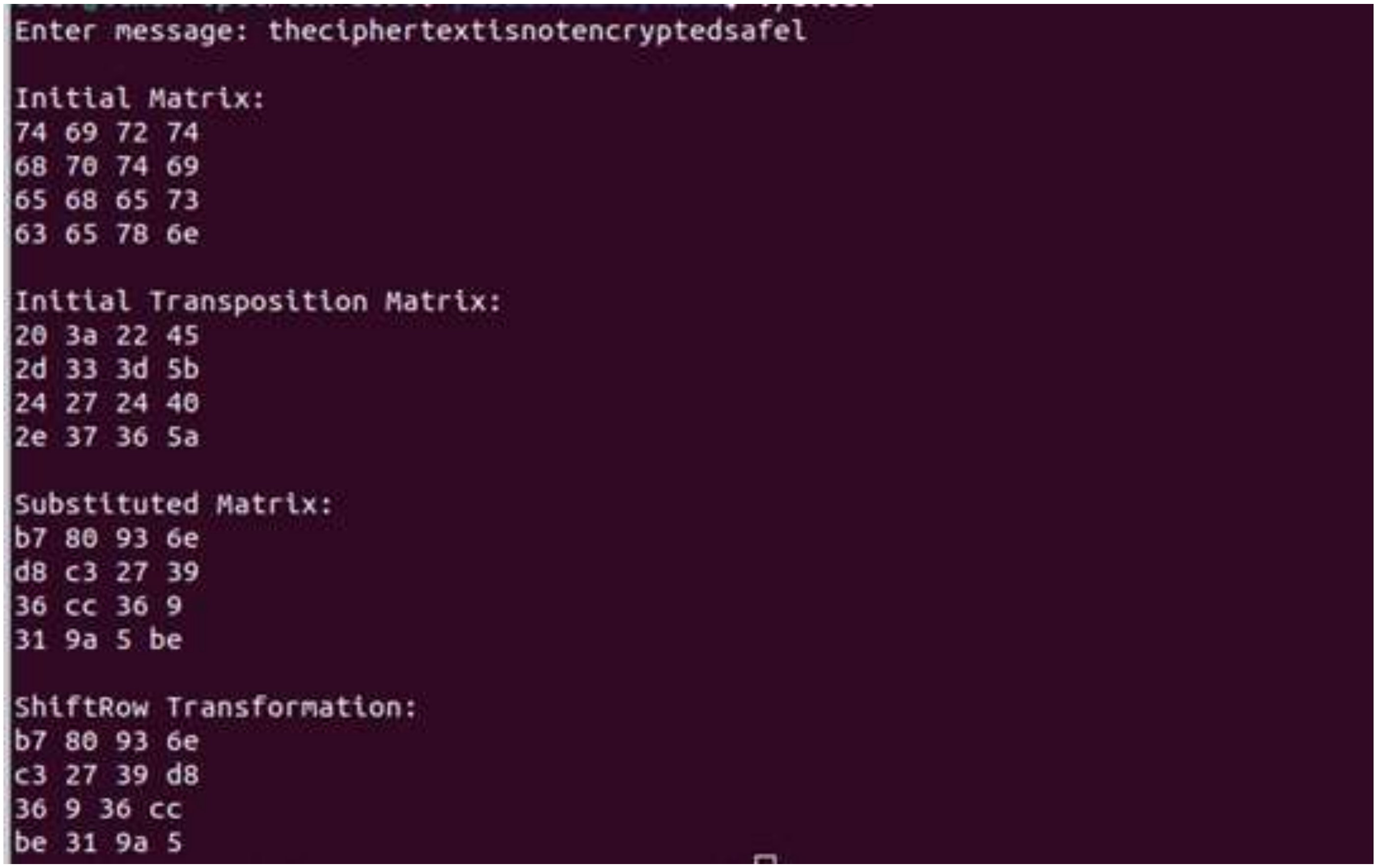
i. Byte substitution using S-Box

ii. ShiftRows using left shift

Algorithm:



```
#include <bits/stdc++.h>
using namespace std;
unsigned long long sbox[16][16] = {
{ 0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe,
0xd7, 0xab, 0x76 },
{ 0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c,
0xa4, 0x72, 0xc0 },
{ 0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71,
0xd8, 0x31, 0x15 },
{ 0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb,
0x27, 0xb2, 0x75 },
{ 0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29,
0xe3, 0x2f, 0x84 },
{ 0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a,
0x4c, 0x58, 0xcf },
{ 0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50,
0x3c, 0x9f, 0xa8 },
{ 0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10,
0xff, 0xf3, 0xd2 },
{ 0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64,
0x5d, 0x19, 0x73 },
{ 0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde,
0x5e, 0x0b, 0xdb },
{ 0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91,
0x95, 0xe4, 0x79 },
{ 0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65,
0x7a, 0xae, 0x08 },
{ 0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b,
0xbd, 0x8b, 0x8a },
{ 0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86,
0xc1, 0x1d, 0x9e },
{ 0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce,
0x55, 0x28, 0xdf },
{ 0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0,
0x54, 0xbb, 0x16 }
};
unsigned long long key[4][4] = {
 {0x54,0x53,0x50,0x31},
 {0x45,0x43,0x49,0x32},
 {0x41,0x4f,0x41,0x33},
 {0x4d,0x52,0x4e,0x34}
};
string XOR(string x, string y){
 string res = "";
 for(int i=0; i<x.length(); i++)
 {
    res += (x[i] == y[i]) ? "0" : "1";
  }
 return res;
}
string SBoxFun(string byte){
string res = "";
int row = bitset<4>( byte.substr(0,4) ).to_ulong();
int col = bitset<4>( byte.substr(4,4) ).to_ulong();
res = bitset<8>(sbox[row][col]).to_string();
return res;
}
int main(){
 string msg;cout << "Enter message: ";
 cin >> msg;
 string hexMsg="";
 stringstream sstream;
 unsigned long long x;
 for(int i=0; msg[i]!='\0';i++){
    int ascii = msg[i];
    sstream.str("");
    sstream << hex<<ascii;
    hexMsg += sstream.str();
 }
 string mat[4][4] , initTrans[4][4], res[4][4], res1[4][4];
 int k=0;
 for(int i=0;i<4;i++){
   for(int j=0;j<4;j++){
      mat[j][i] = hexMsg.substr(i*8+j*2,2);
    }
 }
 cout << "\nInitial Matrix:\n";
 for(int i=0;i<4;i++){
     for(int j=0;j<4;j++){
         cout << mat[i][j] <<" ";
  }
  cout << endl;
 }
 for(int i=0;i<4;i++){
    for(int j=0;j<4;j++){
       unsigned long long val = stoull(mat[i][j], nullptr, 16);
       string temp1 = bitset<8>(val).to_string();
       string temp2 = bitset<8>(key[i][j]).to_string();
       initTrans[i][j] = XOR(temp1,temp2);
     }
  }
 cout << "\nInitial Transposition Matrix:\n";
 for(int i=0;i<4;i++){
    for(int j=0;j<4;j++){
        cout << hex<< bitset<8>(initTrans[i][j]).to_ulong() <<" ";
 }
cout << endl;
}


 for(int i=0;i<4;i++){
   for(int j=0;j<4;j++){
       res[i][j] = SBoxFun(initTrans[i][j]);
   }
 }
 cout << "\nSubstituted Matrix:\n";
 for(int i=0;i<4;i++){
    for(int j=0;j<4;j++){
       cout << hex<< bitset<8>(res[i][j]).to_ulong() <<" ";
  }
 cout << endl;
 }
 for(int i=0;i<4;i++){
     for(int j=0;j<4;j++){
          res1[i][j] = res[i][(j+i)%4];
 }}
 cout << "\nShiftRow Transformation:\n";
 for(int i=0;i<4;i++){
     for(int j=0;j<4;j++){
          cout << hex<< bitset<8>(res1[i][j]).to_ulong()<<" ";
 }
  cout << endl;
 }
 return 0;
}
```



```
Enter message: theciphertextisnotencryptedsafel

Initial Matrix:
74 69 72 74
68 70 74 69
65 68 65 73
63 65 78 6e

Initial Transposition Matrix:
20 3a 22 45
2d 33 3d 5b
24 27 24 40
2e 37 36 5a

Substituted Matrix:
b7 80 93 6e
d8 c3 27 39
36 cc 36 9
31 9a 5 be

ShiftRow Transformation:
b7 80 93 6e
c3 27 39 d8
36 9 36 cc
be 31 9a 5
```

SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMKUR-572103

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CRYPTOGRAPHY AND NETWORK SECURITY LAB (7RCSL01)

Student Name: A N V S Anudeep	USN: 1SI19CS017	Batch No: A1	Date: 16/01/2023

Evaluation:

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

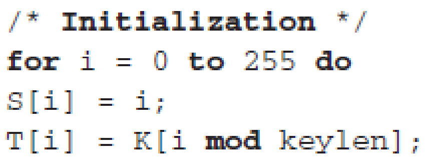
Sl.No	Name of the Faculty In-Charge	Signature
1.	Dr. AS Poornima	
2.	Ravi V	

Question No: 9 Implement the following with respect to RC4:

- **i.** Print first *n* key bytes generated by key generation process.
- **ii.** Illustrate encryption/decryption by accepting one-byte data as input on the above generated keys.

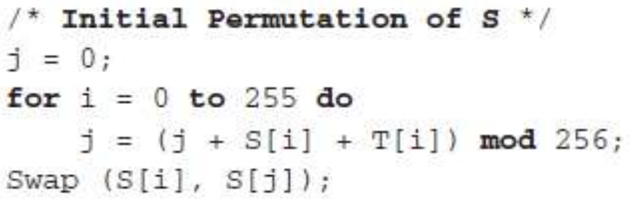
Algorithm: i) To Generate the key, Its three-step process,

Initialization of S:



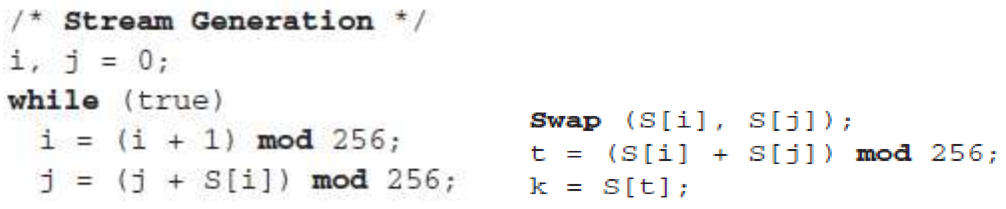
```
/* Initialization */
for i = 0 to 255 do
S[i] = i;
T[i] = K[i mod keylen];
```

Initial Permutation of S:



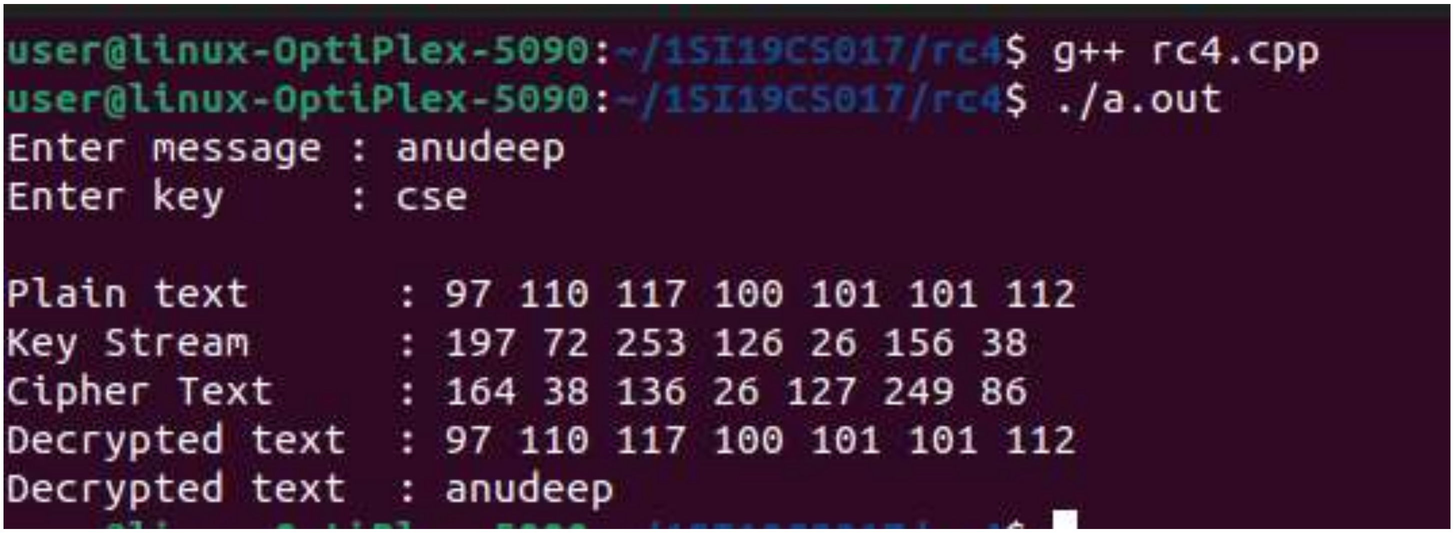
```
/* Initial Permutation of S */
j = 0;
for i = 0 to 255 do
    j = (j + S[i] + T[i]) mod 256;
Swap (S[i], S[j]);
```

Stream Generation:



```
/* Stream Generation */
i, j = 0;
while (true)
  i = (i + 1) mod 256;
  j = (j + S[i]) mod 256;
Swap (S[i], S[j]);
t = (S[i] + S[j]) mod 256;
k = S[t];
```

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

int main()

{
    int S[256], T[256], keyStream[256];
    string ptString, keyString, dtString = "";
    int pt[256], ct[256], dt[256], j;

    cout << "Enter message : "; cin >> ptString;
    cout << "Enter key     : "; cin >> keyString;
    int n = ptString.length();

    cout << "\nPlain text \t: " ;
    for(int i=0; i<n; i++)
    {
        pt[i] = ptString[i];
        cout << pt[i] << " ";
    }


    for(int i=0; i<256; i++)
    {
        S[i] = i;
        T[i] = (int)keyString[i%keyString.length()];
    }


    for(int i=0; i<256; i++)
    {
        j = (j + S[i] + T[i]) % 256;
        swap(S[i], S[j]);
    }


    cout << "\nKey Stream \t: ";
    j=0;
    for(int i=0; i<n; i++)
    {
        j = (j + S[i]) % 256;
        swap(S[i], S[j]);
        int t = (S[i] + S[j]) % 256;
        keyStream[i] = S[t];
        cout << keyStream[i] << " ";
    }


    cout << "\nCipher Text \t: ";
    for(int i=0; i<n; i++)
{
        ct[i] = pt[i] ^ keyStream[i];
        cout << ct[i] << " ";
}



    cout << "\nDecrypted text \t: " ;
    for(int i=0; i<n; i++)
    {
        dt[i] = ct[i] ^ keyStream[i];
        cout << dt[i] << " ";
        dtString += (char)dt[i];
    }
    cout << "\nDecrypted text \t: " << dtString << endl;
}
```



```
user@linux-OptiPlex-5090:~/1SI19CS017/rc4$ g++ rc4.cpp
user@linux-OptiPlex-5090:~/1SI19CS017/rc4$ ./a.out
Enter message : anudeep
Enter key     : cse

Plain text      : 97 110 117 100 101 101 112
Key Stream      : 197 72 253 126 26 156 38
Cipher Text     : 164 38 136 26 127 249 86
Decrypted text  : 97 110 117 100 101 101 112
Decrypted text  : anudeep
```

SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMKUR-572103

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CRYPTOGRAPHY AND NETWORK SECURITY LAB (7RCSL01)

Student Name: ANVS Anudeep	USN: 1SI19CS017	Batch No: A1	Date: 16/01/23

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.	Dr AS Poornima	
2.	Ravi V	

Question No: 10

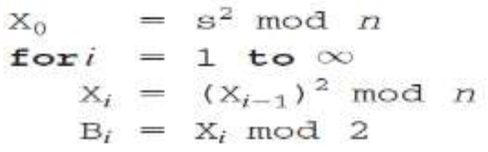
Write a program to generate large random number using BBS random number generator algorithm and check whether the generated number is prime or not using RABIN-MILLER Primality testing algorithm.

Algorithm:

BBS Random Number Generator Algorithm:

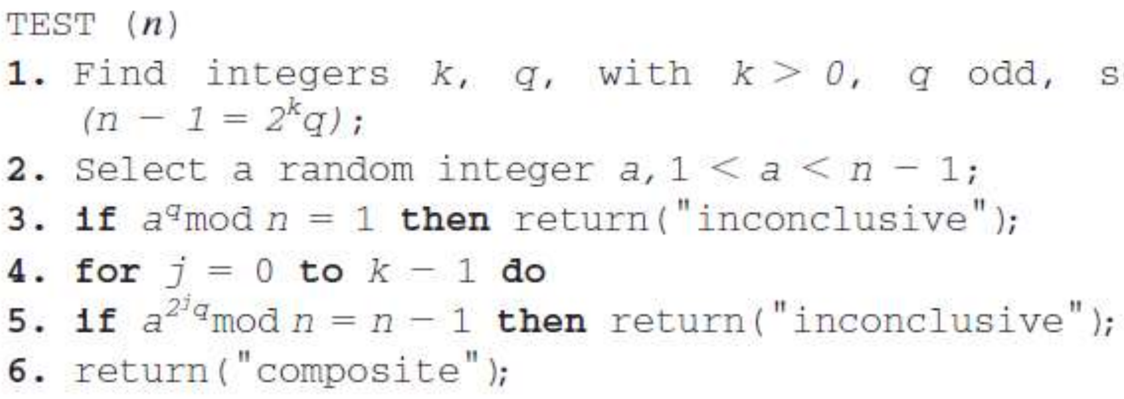
First, choose two large prime numbers p and q, that both have a remainder of 3 when divided by 4.

P=Q=3 mod 4



```
$X_0 = s^2 \bmod n$
for $i = 1$ to $\infty$
    $X_i = (X_{i-1})^2 \bmod n$
    $B_i = X_i \bmod 2$
```

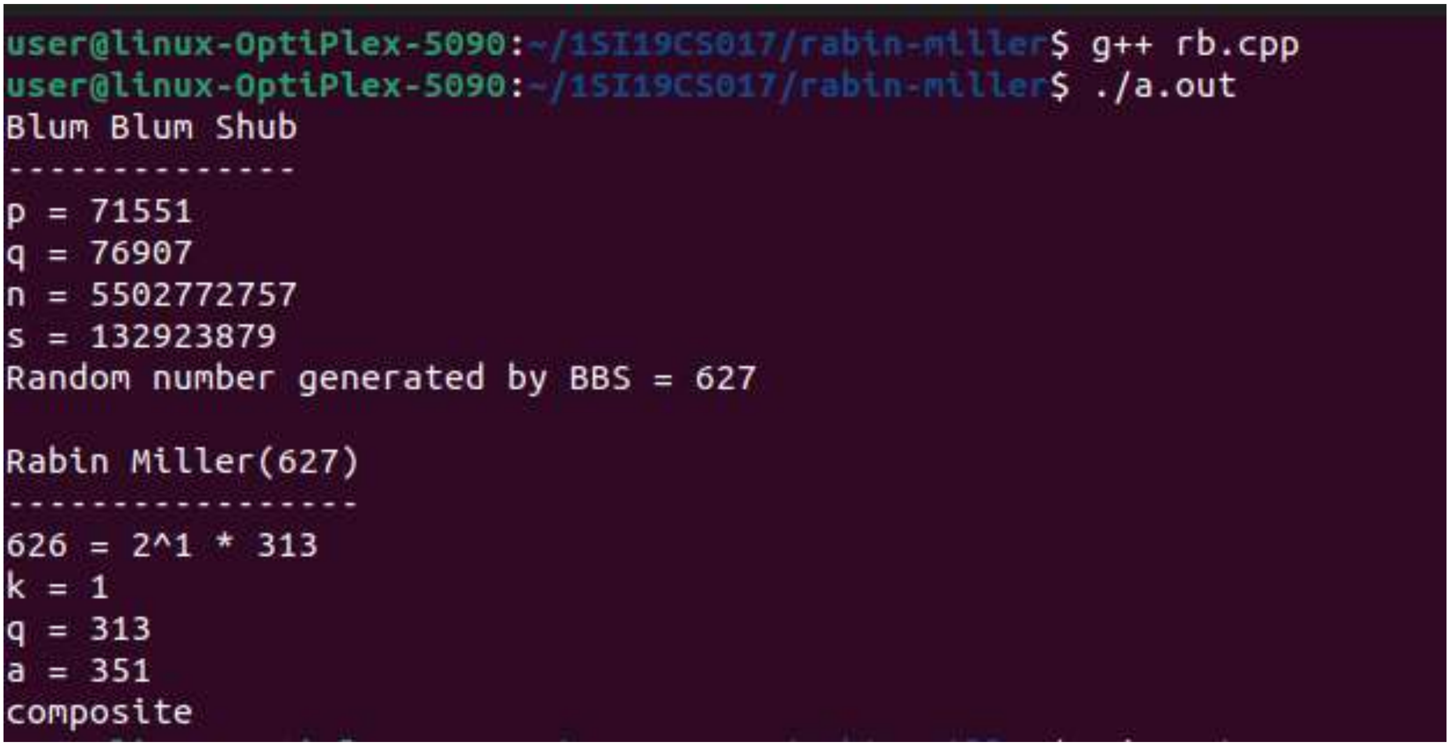
RABIN-MILLER Primality testing algorithm:



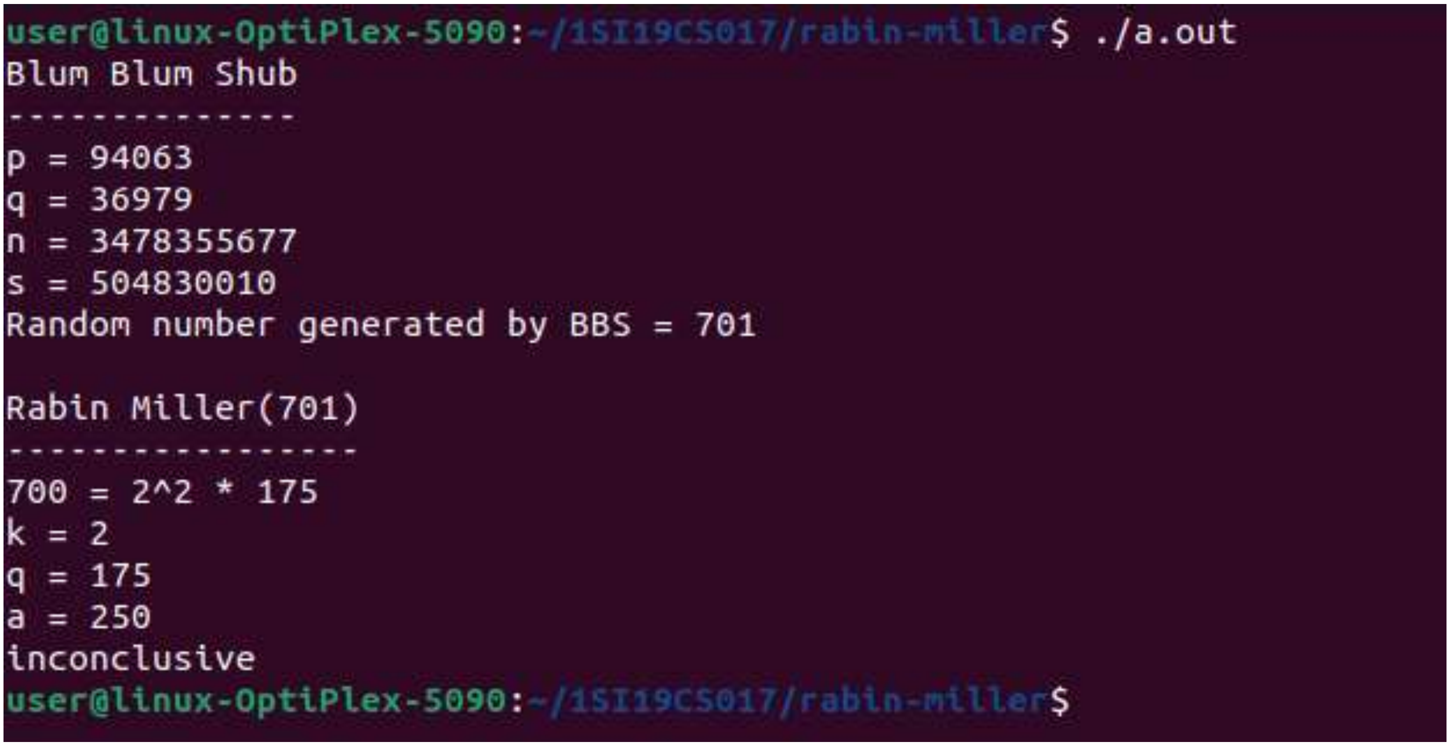
```
TEST ($n$)
1. Find integers $k$, $q$, with $k > 0$, $q$ odd, so that
   $(n - 1 = 2^k q)$;
2. Select a random integer $a, 1 < a < n - 1$;
3. if $a^q \bmod n = 1$ then return("inconclusive");
4. for $j = 0$ to $k - 1$ do
5. if $a^{2^j q} \bmod n = n - 1$ then return("inconclusive");
6. return("composite");
```

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

int randInRange(int low, int high)
{
return rand() % (high-low+1) + (low+1) ;
}

int genPrime3mod4()
{
while(true)
{
int num = randInRange(10000,100000);
if(num%4 != 3) continue;

bool prime = true;
for(int i=2; i<=sqrt(num); i++)
{
if(num % i == 0)
{
prime = false;
break;
}
}
if(prime) return num;
}
}

int bbs(int p, int q)
{
long long n = (long long)p*q ;

long long s;
do{
s = rand();
} while (s%p==0 || s%q==0 || s==0);

int B = 0;
long long x = (s*s) % n;
for(int i=0; i<10; i++)
{
x = (x*x) % n;
B = B<<1 | (x & 1);
}

cout<<"Blum Blum Shub"<<endl<<"--------------"<<endl;
cout<<"p = "<< p <<"\nq = "<< q <<"\nn = "<< n <<"\ns = "<< s <<endl;
return B;
}
```

```
int powModN(int a, int b, int n)
{
int res=1;
for(int i=0; i<b; i++)
{
res = (res * a) % n;
}
return res;
}

string rabinMiller(int n)
{
int k = 0;
int q = n-1;
while(q % 2 == 0)
{
q = q/2 ;
k++ ;
}

int a = randInRange(1, n-1);

cout << "\nRabin Miller(" << n << ")\n------------------" << endl;
cout << n-1 << " = 2^" << k << " * " << q << endl;
cout << "k = " << k << "\nq = " << q << "\na = " << a << endl;

if(powModN(a,q,n) == 1) return "inconclusive";

for(int j=0; j<k ; j++)
{
if(powModN(a, pow(2,j)*q, n) == n-1) return "inconclusive";
}
return "composite";
}

int main()
{
srand(time(NULL));
int p = genPrime3mod4();
int q = genPrime3mod4();
int randNum = bbs(p, q);
cout << "Random number generated by BBS = " << randNum << endl;

cout << rabinMiller(randNum) << endl ;

return 0;
}
```



```
user@linux-OptiPlex-5090:~/1SI19CS017/rabin-miller$ g++ rb.cpp
user@linux-OptiPlex-5090:~/1SI19CS017/rabin-miller$ ./a.out
Blum Blum Shub
--------------
p = 71551
q = 76907
n = 5502772757
s = 132923879
Random number generated by BBS = 627

Rabin Miller(627)
-----------------
626 = 2^1 * 313
k = 1
q = 313
a = 351
composite
```



```
user@linux-OptiPlex-5090:~/1SI19CS017/rabin-miller$ ./a.out
Blum Blum Shub
--------------
p = 94063
q = 36979
n = 3478355677
s = 504830010
Random number generated by BBS = 701

Rabin Miller(701)
-----------------
700 = 2^2 * 175
k = 2
q = 175
a = 250
inconclusive
user@linux-OptiPlex-5090:~/1SI19CS017/rabin-miller$
```

SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMKUR-572103

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CRYPTOGRAPHY AND NETWORK SECURITY LAB (7RCSL01)

Student Name: ANVS Anudeep	USN: 1SI19CS017	Batch No: A1	Date: 23/01/2023

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.	Dr AS Poornima	
2.	Ravi V	

Question No: 11

Implement RSA algorithm to process blocks of plaintext (refer Figure 9.7 of the text book), where plaintext is a string of characters and let the block size be two characters. (Note: assign a unique code to each plain text character i.e., a=00, A=26). The program should support the following.

i. Accept string of characters as plaintext.
ii. Encryption takes plaintext and produces ciphertext characters
iii. Decryption takes ciphertext characters obtained in step ii and produces corresponding plaintext characters.
iv. Display the result after each step

Algorithm:

1. Generate *e*,*p*,*q* using random number generator.
2. Calculate n value , n=p×q.
3. Determine public and private keys *(e,n) and (d,n).*
4. Accept plain text in string format and assign numbers between 0 to 26 for characters (a to z)
5. Plain text in decimal string {p1,p2,p3….} is encrypted using public key as shown in fig 1.

$$C_1 = P_1^{\,e} \bmod n$$
$$C_2 = P_2^{\,e} \bmod n$$
$$\vdots$$

Fig 1

Recovered decimal text

$$P_1 = C_1^{\,d} \bmod n$$
$$P_2 = C_2^{\,d} \bmod n$$
$$\vdots$$

Fig 2.

6. Transmit the cipher text in decimal format to server using through sockets for decryption.

Server should decrypt the cipher text {c1,c2,c3…} shown in fig 2. and print the string in character format back to screen.

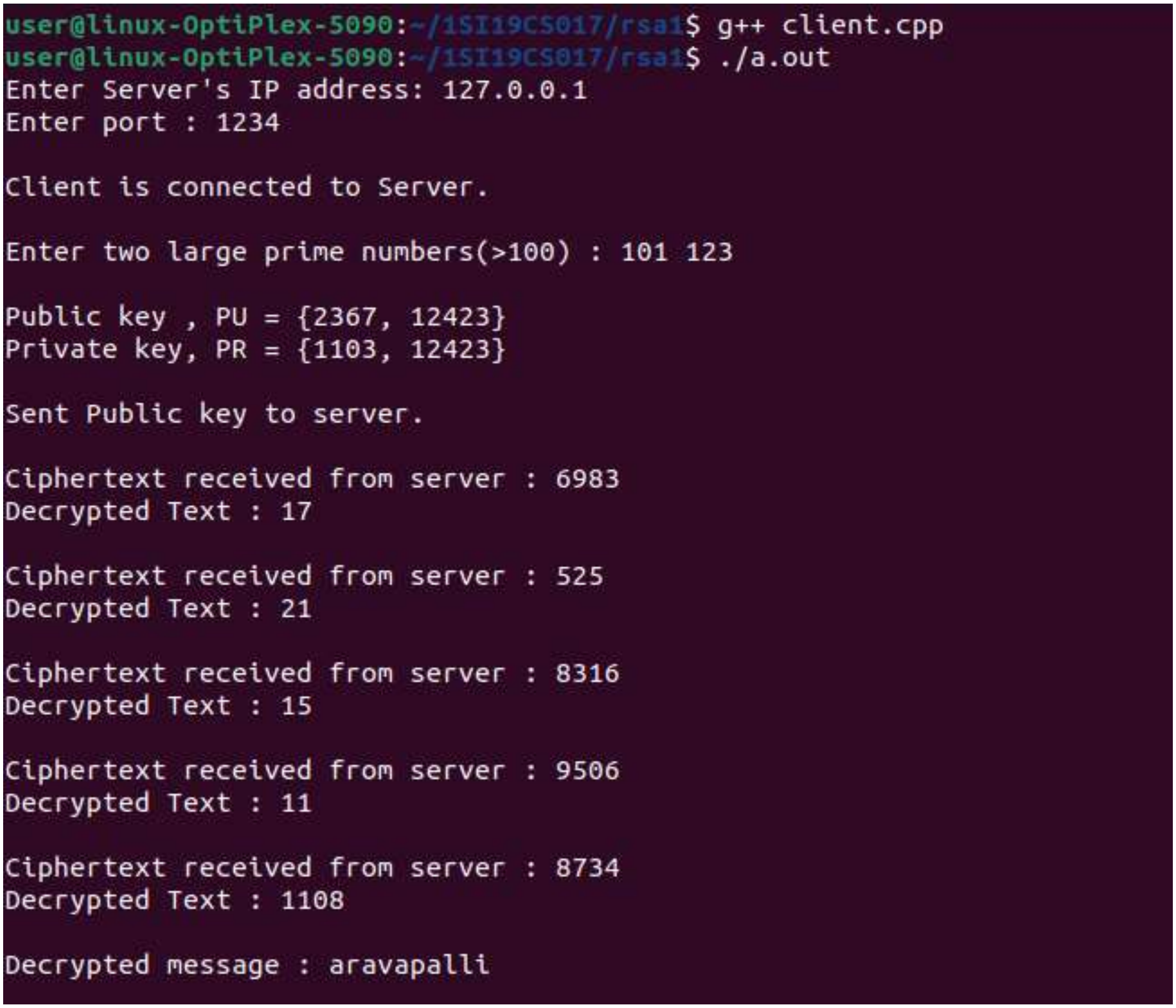
Client

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

int connectToServer(const char* ip, int port)
{
    int sock = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in addr = {AF_INET, htons(port), inet_addr(ip)};

    if(connect(sock, (struct sockaddr *) &addr, sizeof(addr)) < 0 ){
        cout << "\nRun server program first." << endl; exit(0);
    }else{
        cout << "\nClient is connected to Server." << endl;
    }
    return sock;
}

int randInRange(int low, int high)
{
    return rand()%(high-(low+1)) + (low+1) ;
}

int gcd(int a, int b)
{
    return b==0 ? a : gcd(b, a%b);
}

int powermod(int a, int b, int n)
{
    int res = 1;
    for(int i=0; i<b; i++)
    {
        res = (res*a) % n;
    }
    return res;
}

int decrypt(int C, int PR[2])
{
    return powermod(C, PR[0], PR[1]);
}

char toChar(int n)
{
    return (n >= 26) ? (n+'A'-26) : (n+'a');
}
```

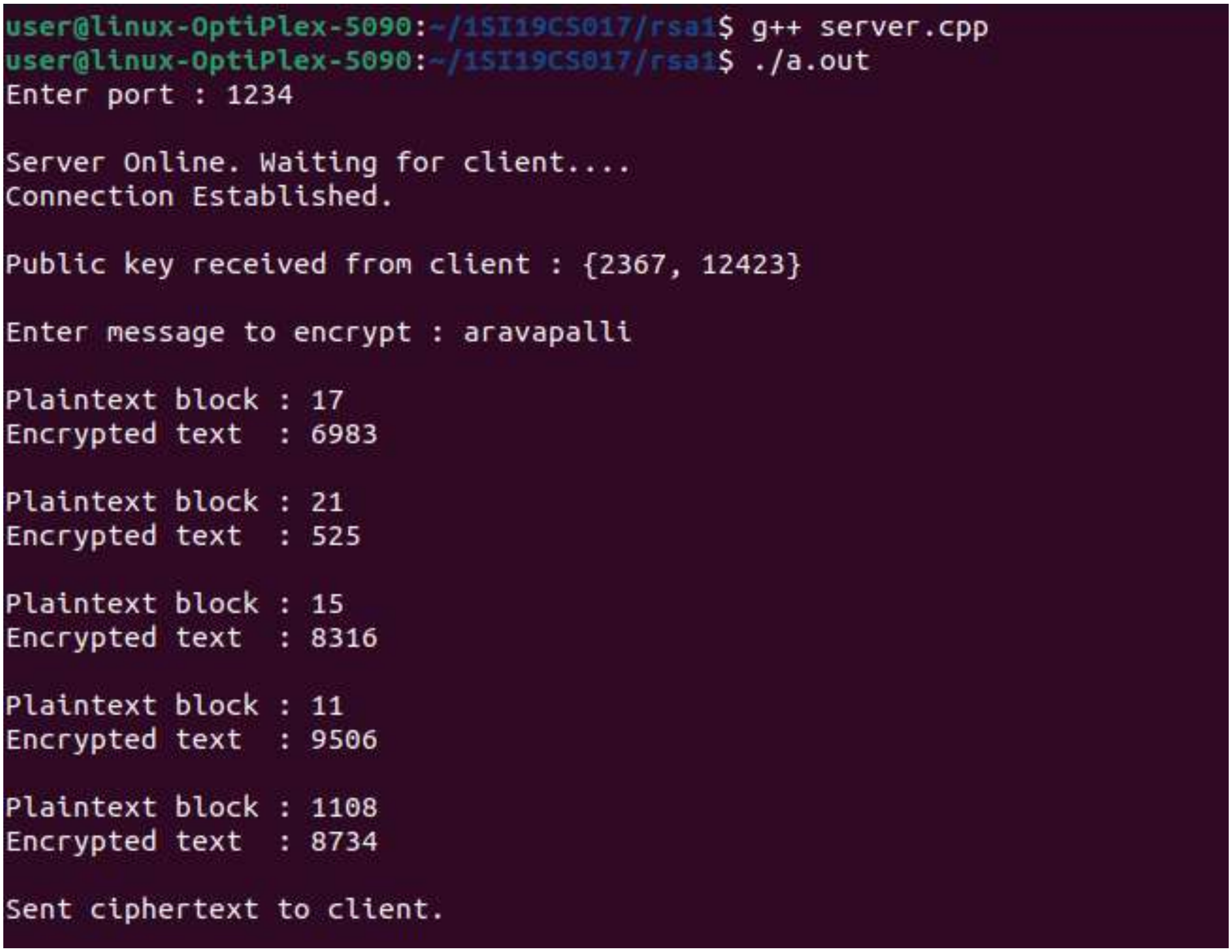
```
int main()

{
    char ip[50];
    int port;
    cout << "Enter Server's IP address: "; cin >> ip;
    cout << "Enter port : "; cin >> port;
    int sock = connectToServer(ip, port);

    int p,q;
    cout << "\nEnter two large prime numbers(>100) : "; cin >> p >> q;
    int n = p * q ;
    int phi = (p-1) * (q-1);

    srand(time(NULL));
    int e, d;
    do{ e = randInRange(1, phi); } while(gcd(e,phi) != 1);

    for(d=1; d<phi; d++)
    {
        if((d*e)%phi == 1) break;
    }

    int PU[2] = {e, n};
    int PR[2] = {d, n};
    cout << "\nPublic key , PU = {" << e << ", " << n << "}" << endl;
    cout << "Private key, PR = {" << d << ", " << n << "}" << endl;

    send(sock, &PU, sizeof(PU), 0);
    cout << "\nSent Public key to server." << endl;

    string msg = "";
    while (true)
    {
        int C;
        recv(sock, &C, sizeof(C), 0);
        if(C == -1)   break;
        cout << "\nCiphertext received from server : " << C << endl;

        int M = decrypt(C,PR);
        cout << "Decrypted Text : " << M << endl;
        msg += toChar(M/100);
        msg += toChar(M%100);
    }
    cout << "\nDecrypted message : " << msg << endl << endl;
}
```

Server

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

int createServer(int port)
{
    int sersock = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in addr = {AF_INET, htons(port), INADDR_ANY};

    bind(sersock, (struct sockaddr *) &addr, sizeof(addr));
    cout << "\nServer Online. Waiting for client...." << endl;

    listen(sersock, 5);
    int sock = accept(sersock, NULL, NULL);
    cout << "Connection Established." << endl;

    return sock;
}

int powermod(int a, int b, int n)
{
    int res = 1;
    for(int i=0; i<b; i++)
    {
        res = (res*a) % n;
    }
    return res;
}


int encrypt(int M, int PU[2])
{
    return powermod(M, PU[0], PU[1]);
}

int toInt(char c)
{
    return (c < 'a') ? (c-'A'+26) : (c-'a');
}

int main()
{
    int port;
    cout << "Enter port : "; cin >> port;
    int sock = createServer(port);

    int PU[2];
    recv(sock, &PU, sizeof(PU), 0);
    cout << "\nPublic key received from client : {" << PU[0] << ", " << PU[1] << "}" << endl;
    string msg;
    cout << "\nEnter message to encrypt : "; cin >> msg;

    if(msg.length()% 2 != 0) msg+="x";

    for(int i=0; i<msg.length(); i+=2)
    {
        int M = toInt(msg[i])*100 + toInt(msg[i+1]);
        cout << "\nPlaintext block : " << M << endl;

        int C = encrypt(M, PU);
        cout << "Encrypted text  : " << C << endl;
        send(sock, &C, sizeof(C), 0);
    }
    int stop = -1;
    send(sock, &stop, sizeof(stop), 0);
    cout << "\nSent ciphertext to client." << endl << endl;
}
```



```
user@linux-OptiPlex-5090:~/1SI19CS017/rsa1$ g++ client.cpp
user@linux-OptiPlex-5090:~/1SI19CS017/rsa1$ ./a.out
Enter Server's IP address: 127.0.0.1
Enter port : 1234

Client is connected to Server.

Enter two large prime numbers(>100) : 101 123

Public key , PU = {2367, 12423}
Private key, PR = {1103, 12423}

Sent Public key to server.

Ciphertext received from server : 6983
Decrypted Text : 17

Ciphertext received from server : 525
Decrypted Text : 21

Ciphertext received from server : 8316
Decrypted Text : 15

Ciphertext received from server : 9506
Decrypted Text : 11

Ciphertext received from server : 8734
Decrypted Text : 1108

Decrypted message : aravapalli
```



```
user@linux-OptiPlex-5090:~/1SI19CS017/rsa1$ g++ server.cpp
user@linux-OptiPlex-5090:~/1SI19CS017/rsa1$ ./a.out
Enter port : 1234

Server Online. Waiting for client....
Connection Established.

Public key received from client : {2367, 12423}

Enter message to encrypt : aravapalli

Plaintext block : 17
Encrypted text  : 6983

Plaintext block : 21
Encrypted text  : 525

Plaintext block : 15
Encrypted text  : 8316

Plaintext block : 11
Encrypted text  : 9506

Plaintext block : 1108
Encrypted text  : 8734

Sent ciphertext to client.
```

SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMKUR-572103

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CRYPTOGRAPHY AND NETWORK SECURITY LAB (7RCSL01)

Student Name: ANVS Anudeep	**USN: 1SI19CS017**	**Batch No: A1**	**Date: 23/01/2023**

Evaluation:

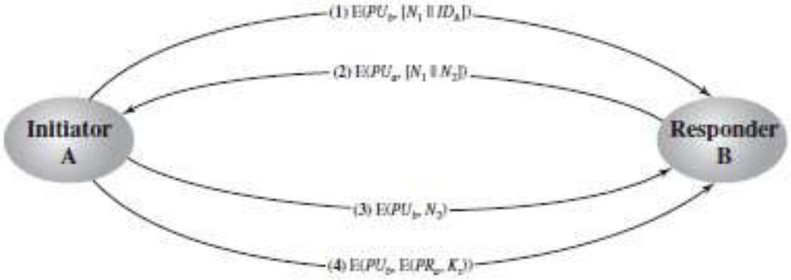
Write Up **(10 marks)**	**Clarity in concepts** **(10 marks)**	**Implementation and execution of the algorithms (10 marks)**	**Viva** **(05 marks)**	**Total** **(35 marks)**

Sl.No	**Name of the Faculty In-Charge**	**Signature**
1.	**Dr AS Poornima**	
2.	**Ravi V**	

Question No: 12

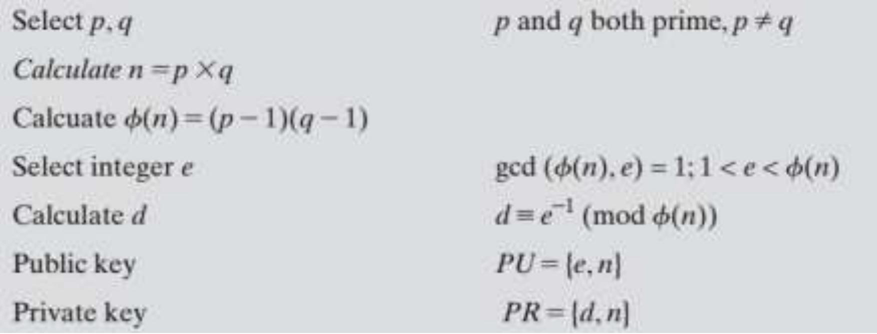
Implement RSA algorithm using client-server concept. Using this illustrate secret key distribution scenario with confidentiality and authentication. The program should support the following :

- i. Both client and server generate {PU, PR} and distribute PU to each other.
- ii. Establish a secret key K between client and server by exchanging the messages as shown in below figure.


(1) $E(PU_b, [N_1 \| ID_A])$
(2) $E(PU_a, [N_1 \| N_2])$
Initiator A
Responder B
(3) $E(PU_b, N_2)$
(4) $E(PU_b, E(PR_a, K_s))$

Algorithm:

- i. Both client and server generate {PU, PR} and distribute PU to each other.



Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calcuate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

- ii. Establish a secret key K between client and server by exchanging the messages as shown in below figure.

Client

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


int p, q, e, d, n, phi;
int PUc[2], PRc[2];
int PUs[2];
int sock;

void connectToServer(const char* ip, int port)
{
   sock = socket(AF_INET, SOCK_STREAM, 0);
   struct sockaddr_in addr = {AF_INET, htons(port), inet_addr(ip)};
   if(connect(sock, (struct sockaddr *) &addr, sizeof(addr)) < 0 ){
      cout << "\nRun server program first." << endl; exit(0);
   }else{
      cout << "\nClient is connected to Server." << endl;
   }
}

int randInRange(int low, int high)
{
   return rand()%(high-(low+1)) + (low+1) ;
}

int gcd(int a, int b)
{
   return b==0 ? a : gcd(b, a%b);
}

void genKey()
{
   cout << "\nEnter two prime numbers (>100) : "; cin >> p >> q;
   n = p * q ;
   phi = (p-1) * (q-1);

   srand(time(NULL));
   do{ e = randInRange(1, phi); } while(gcd(e,phi) != 1);
   for(d=1; d<phi; d++)
   {
      if((d*e)%phi == 1) break;
   }

   PUc[0] = e; PUc[1] = n;
   PRc[0] = d; PRc[1] = n;
   cout << "\nPublic key , PUc = {" << e << ", " << n << "}" << endl;
   cout <<   "Private key, PRc = {" << d << ", " << n << "}" << endl;
}


void shareKey()
{
   recv(sock, &PUs, sizeof(PUs), 0);
   send(sock, &PUc, sizeof(PUc), 0);
   cout << "Public key received from server, PUs = {" << PUs[0] << ", " << PUs[1] << "}" << endl;
   cout << "\nSent client's Public key to server." << endl;
}

int powermod(int a, int b, int n)
{
   int res = 1;
   for(int i=0; i<b; i++)
   {
      res = (res*a) % n;
   }
   return res;
}

int encrypt(int M, int PU[2])
{
   return powermod(M, PU[0], PU[1]);
}

int decrypt(int C, int PR[2])
{
   return powermod(C, PR[0], PR[1]);
}

int main()
{
   char ip[50];  cout<<"\nEnter server's IP address: "; cin>>ip;
   int port;     cout<<"Enter port : ";  cin>>port;
   srand(time(NULL));

   connectToServer(ip, port);
   genKey();
   shareKey();

   int cipher;
   recv(sock, &cipher, sizeof(cipher), 0);
   cout << "\nReceived encrypted (N1||ID) from server : " << cipher << endl;
   int msg = decrypt(cipher, PRc);
   int N1 = msg/100;
   int ID = msg%100;
   cout << "Decrypted Server's ID,   IDs = " << ID << endl;
   cout << "Decrypted Server's nonce, N1 = " << N1 << endl;

   int N2 = rand() % 100;
   cout << "\nNonce generated, N2 = " << N2 << endl;
   msg = N1*100 + N2;



   cipher = encrypt(msg, PUs);
   send(sock, &cipher, sizeof(cipher), 0);
   cout << "Sent encrypted (N1||N2) to server : " << cipher << endl;
```

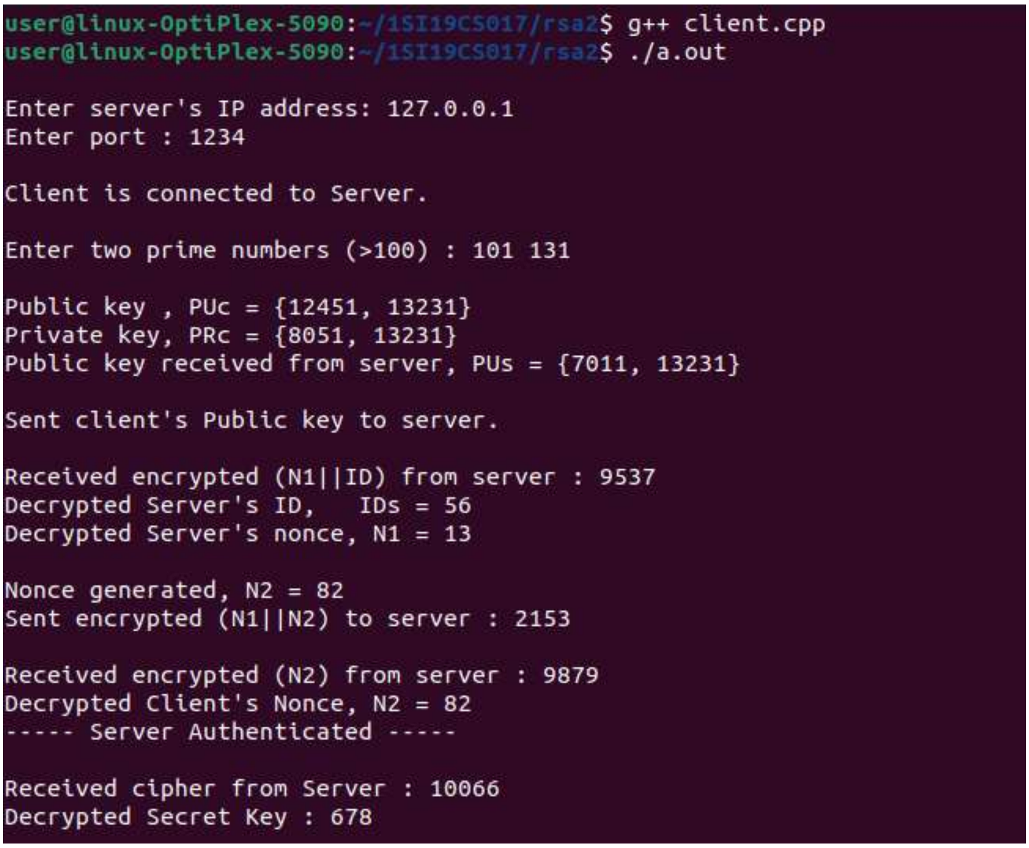
```
    recv(sock, &cipher, sizeof(cipher), 0);
    cout << "\nReceived encrypted (N2) from server : " << cipher << endl;
    int N2s = decrypt(cipher, PRc);
    cout << "Decrypted Client's Nonce, N2 = " << N2s << endl;
    if(N2s != N2) {cout << "\nNonce didn't match!\n"; exit(-1);}
    else {cout << "----- Server Authenticated -----" << endl;}

    int k;
    recv(sock, &cipher, sizeof(cipher), 0);
    cout << "\nReceived cipher from Server : " << cipher << endl;
    k = decrypt(decrypt(cipher, PRc), PUs);
    cout << "Decrypted Secret Key : " << k << endl << endl;
}
```

Server

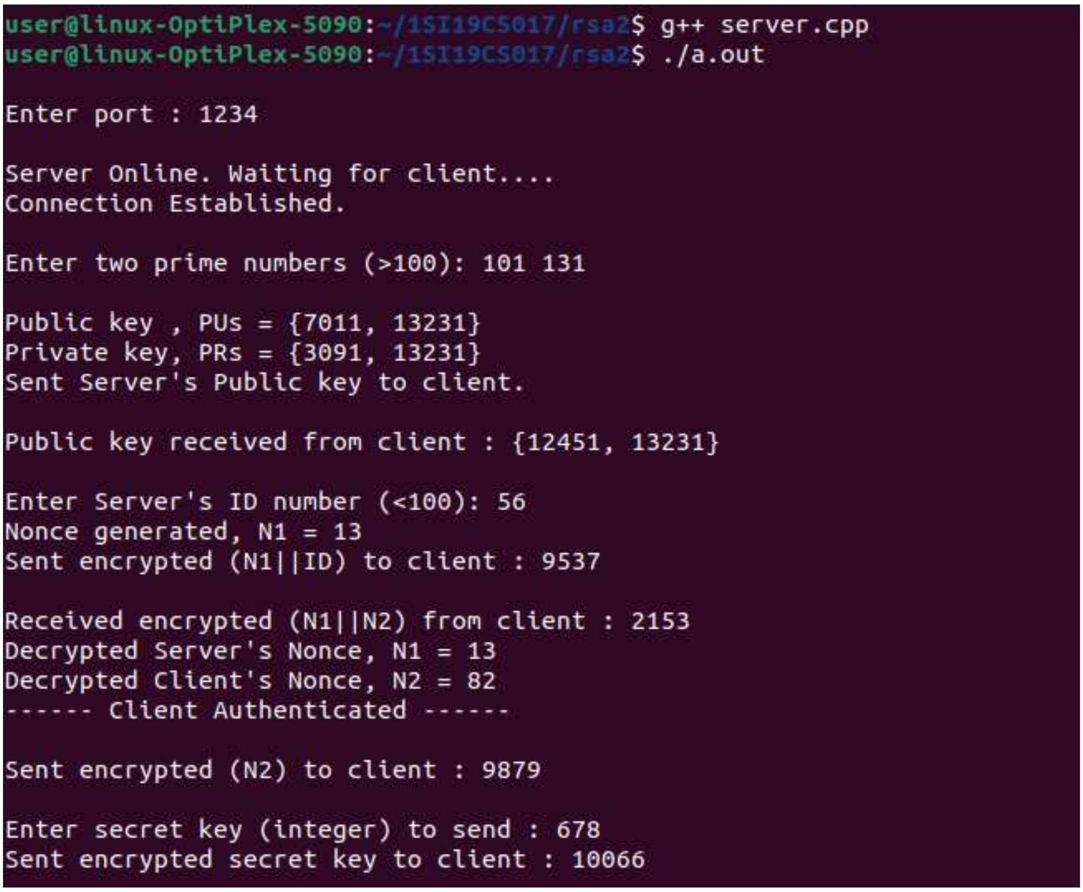
```
# include <bits/stdc++.h>
# include <arpa/inet.h>
using namespace std;

int p, q, e, d, n, phi;
int PUs[2], PRs[2];
int PUc[2];
int sock;

void createServer(int port)
{
    int sersock = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in addr = {AF_INET, htons(port), INADDR_ANY};

    bind(sersock, (struct sockaddr *) &addr, sizeof(addr));
    cout << "\nServer Online. Waiting for client...." << endl;

    listen(sersock, 5);
    sock = accept(sersock, NULL, NULL);
    cout << "Connection Established." << endl;
}

int randInRange(int low, int high)
{
    return rand()%(high-(low+1)) + (low+1) ;
}

int gcd(int a, int b)
{



    return b==0 ? a : gcd(b, a%b);
}

void genKey()
{
    cout << "\nEnter two prime numbers (>100): "; cin >> p >> q;
```

```
   n = p * q ;
   phi = (p-1) * (q-1);

   srand(time(NULL));
   do{ e = randInRange(1, phi); } while(gcd(e,phi) != 1);
   for(d=1; d<phi; d++)
   {
      if((d*e)%phi == 1) break;
   }

   PUs[0] = e; PUs[1] = n;
   PRs[0] = d; PRs[1] = n;
   cout << "\nPublic key , PUs = {" << e << ", " << n << "}" << endl;
   cout <<   "Private key, PRs = {" << d << ", " << n << "}" << endl;
}

void shareKey()
{
   send(sock, &PUs, sizeof(PUs), 0);
   recv(sock, &PUc, sizeof(PUc), 0);
   cout << "Sent Server's Public key to client." << endl;
   cout << "\nPublic key received from client : {" << PUc[0] << ", " << PUc[1] << "}" << endl;
}

int powermod(int a, int b, int n)
{
   int res = 1;
   for(int i=0; i<b; i++)
   {
      res = (res*a) % n;
   }
   return res;
}

int encrypt(int M, int PU[2])
{
   return powermod(M, PU[0], PU[1]);
}

int decrypt(int C, int PR[2])
{
   return powermod(C, PR[0], PR[1]);
}




int main()
{
   int port;  cout<<"\nEnter port : ";  cin>>port;
   srand(time(NULL));

   createServer(port);
   genKey();
   shareKey();
```

```
    int ID;  cout<<"\nEnter Server's ID number (<100): "; cin>>ID;
    int N1 = rand()%100;
    cout << "Nonce generated, N1 = " << N1 << endl;

    int msg = N1*100 + ID;
    int cipher = encrypt(msg, PUc);
    send(sock, &cipher, sizeof(cipher), 0);
    cout << "Sent encrypted (N1||ID) to client : " << cipher << endl;

    recv(sock, &cipher, sizeof(cipher), 0);
    cout << "\nReceived encrypted (N1||N2) from client : " << cipher << endl;
    msg = decrypt(cipher, PRs);
    int N1c = msg/100;
    int N2 = msg%100;
    cout << "Decrypted Server's Nonce, N1 = " << N1c << endl;
    cout << "Decrypted Client's Nonce, N2 = " << N2 << endl;
    if(N1 != N1c) {cout << "\nNonce didn't match!\n"; exit(-1);}
    else {cout << "------ Client Authenticated ------" << endl;}

    cipher = encrypt(N2, PUc);
    send(sock, &cipher, sizeof(cipher), 0);
    cout << "\nSent encrypted (N2) to client : " << cipher << endl;

    int k;
    cout << "\nEnter secret key (integer) to send : "; cin >> k;
    cipher = encrypt(encrypt(k,PRs), PUc);
    send(sock, &cipher, sizeof(cipher), 0);
    cout << "Sent encrypted secret key to client : " << cipher << endl << endl;
}
```



```
user@linux-OptiPlex-5090:~/1SI19CS017/rsa2$ g++ client.cpp
user@linux-OptiPlex-5090:~/1SI19CS017/rsa2$ ./a.out

Enter server's IP address: 127.0.0.1
Enter port : 1234

Client is connected to Server.

Enter two prime numbers (>100) : 101 131

Public key , PUc = {12451, 13231}
Private key, PRc = {8051, 13231}
Public key received from server, PUs = {7011, 13231}

Sent client's Public key to server.

Received encrypted (N1||ID) from server : 9537
Decrypted Server's ID,   IDs = 56
Decrypted Server's nonce, N1 = 13

Nonce generated, N2 = 82
Sent encrypted (N1||N2) to server : 2153

Received encrypted (N2) from server : 9879
Decrypted Client's Nonce, N2 = 82
----- Server Authenticated -----

Received cipher from Server : 10066
Decrypted Secret Key : 678
```



```
user@linux-OptiPlex-5090:~/1SI19CS017/rsa2$ g++ server.cpp
user@linux-OptiPlex-5090:~/1SI19CS017/rsa2$ ./a.out

Enter port : 1234

Server Online. Waiting for client....
Connection Established.

Enter two prime numbers (>100): 101 131

Public key , PUs = {7011, 13231}
Private key, PRs = {3091, 13231}
Sent Server's Public key to client.

Public key received from client : {12451, 13231}

Enter Server's ID number (<100): 56
Nonce generated, N1 = 13
Sent encrypted (N1||ID) to client : 9537

Received encrypted (N1||N2) from client : 2153
Decrypted Server's Nonce, N1 = 13
Decrypted Client's Nonce, N2 = 82
------ Client Authenticated ------

Sent encrypted (N2) to client : 9879

Enter secret key (integer) to send : 678
Sent encrypted secret key to client : 10066
```

SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMKUR-572103

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CRYPTOGRAPHY AND NETWORK SECURITY LAB (7RCSL01)

Student Name: ANVS Anudeep	**USN: 1SI19CS017**	**Batch No: A1**	**Date: 23/01/2023**

Evaluation:

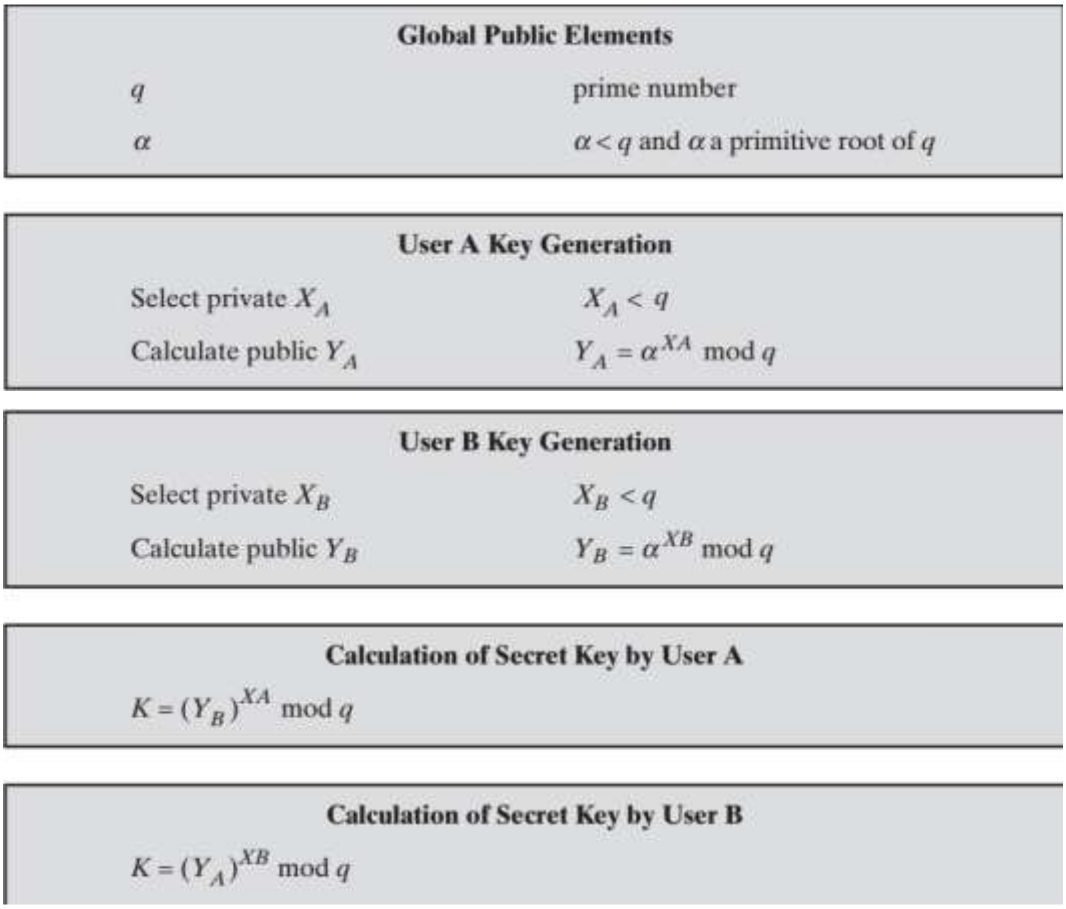
Write Up (10 marks)	Clarity in concepts (10 marks)	Implementation and execution of the algorithms (10 marks)	Viva (05 marks)	Total (35 marks)

Sl.No	Name of the Faculty In-Charge	Signature
1.	**Dr AS Poornima**	
2.	**Ravi V**	

Question No: 13

Compute common secret key between client and server using Diffie-Hellman key exchange technique. Perform encryption and decryption of message using the shared secret key **(Use simple XOR operation to encrypt and decrypt the message.)**

Algorithm:



Global Public Elements

q	prime number
α	$\alpha < q$ and α a primitive root of q

User A Key Generation

Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{XA} \bmod q$

User B Key Generation

Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{XB} \bmod q$

Calculation of Secret Key by User A

$K = (Y_B)^{XA} \bmod q$

Calculation of Secret Key by User B

$K = (Y_A)^{XB} \bmod q$

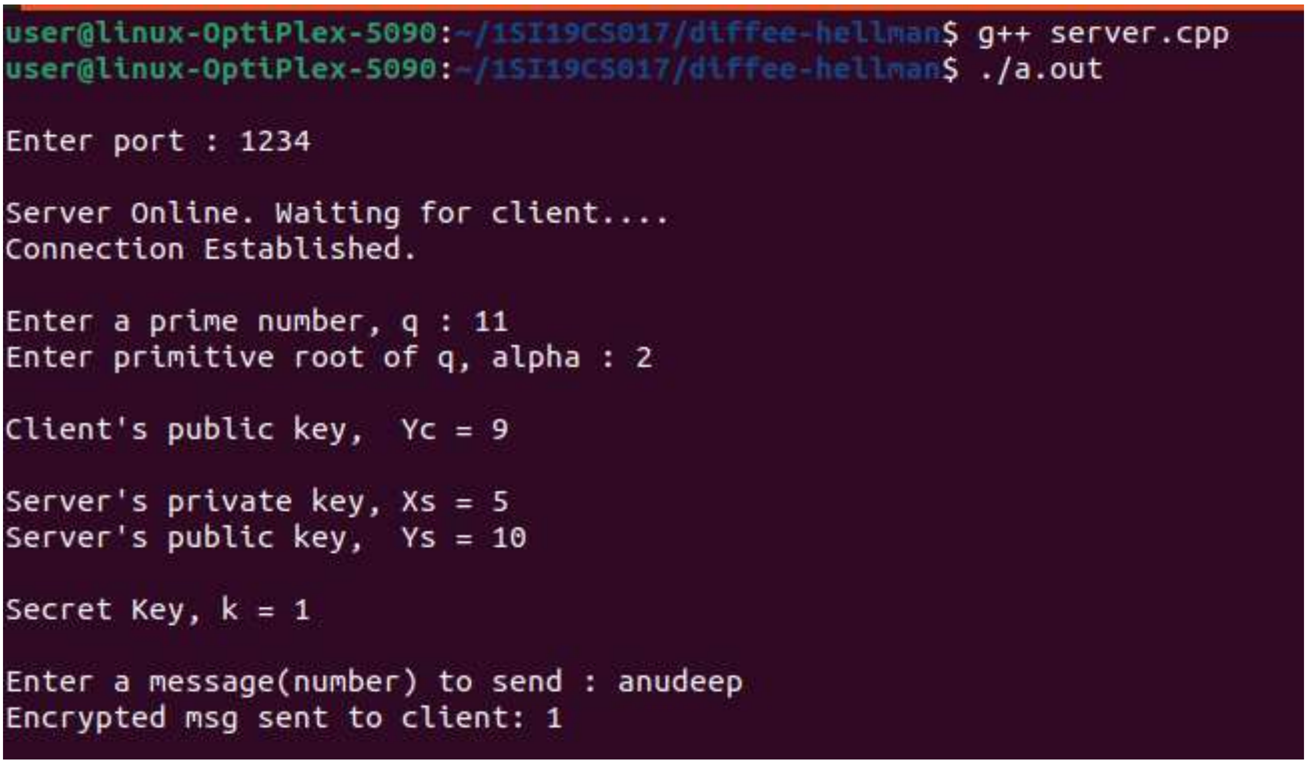
Client

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

int connectToServer(const char* ip, int port)
{
   int sock = socket(AF_INET, SOCK_STREAM, 0);
   struct sockaddr_in addr = {AF_INET, htons(port),inet_addr(ip)};

   if(connect(sock, (struct sockaddr *) &addr, sizeof(addr)) < 0 ){
       cout << "\nRun server program first." << endl; exit(0);
   }else{
       cout << "\nClient is connected to Server." << endl;
   }
   return sock;
}

int randInRange(int low, int high)
{
   return rand()%(high-(low+1)) + (low+1) ;
}

long powermod(long a, long b, long  q)
{
long res=1;
for(long i=0;i<b;i++)
{
    res=(res*a)%q;
}
return res;
}

int main()
{
   char ip[50]; cout << "\nEnter server's IP address: "; cin >> ip;
   int port;    cout << "Enter port : "; cin >> port;
   int sock = connectToServer(ip, port);

   long q, alpha;
   cout<<"\nEnter a prime number, q : "; cin >> q;
   cout<<"Enter primitive root of q, alpha : "; cin >> alpha;

   srand(time(NULL));
   long Xc = randInRange(1, q);
   cout<< "\nClient's private key, Xc = " << Xc <<endl;

   long Yc = powermod(alpha, Xc, q);
   send(sock, &Yc, sizeof(Yc), 0);
   cout<< "Client's public key,  Yc = " << Yc <<endl;
```

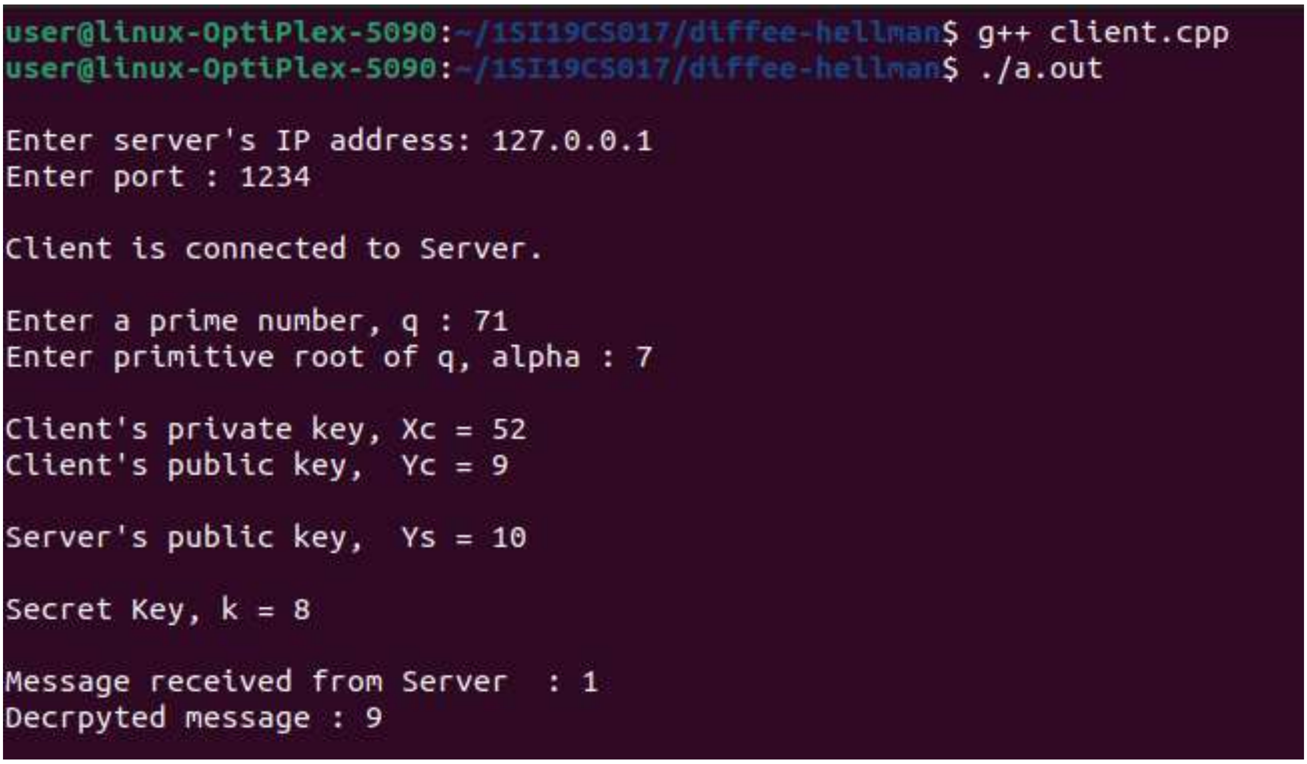
```
 long Ys;
 recv(sock, &Ys, sizeof(Ys), 0);
 cout<< "\nServer's public key,  Ys = " << Ys <<endl;

 long k = powermod(Ys,Xc,q);
 cout<<"\nSecret Key, k = "<<k<<endl;

 long cipher;
 recv(sock, &cipher, sizeof(cipher), 0);
 cout<<"\nMessage received from Server  : " << cipher << endl;

 long decipher = cipher ^ k;
 cout << "Decrpyted message : " << decipher << endl << endl;
 }
```

server

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

int createServer(int port)
{
   int sersock = socket(AF_INET, SOCK_STREAM, 0);
   struct sockaddr_in addr = {AF_INET, htons(port), INADDR_ANY};

   bind(sersock, (struct sockaddr *) &addr, sizeof(addr));
   cout << "\nServer Online. Waiting for client...." << endl;

   listen(sersock, 5);
   int sock = accept(sersock, NULL, NULL);
   cout << "Connection Established." << endl;
   return sock;
}

 int randInRange(int low, int high)
{
    return rand()%(high-(low+1)) + (low+1) ;
}

long powermod(long a, long b, long  q)
{
   long res=1;
   for(long i=0; i<b; i++)
  {
      res=(res*a)%q;
  }
 return res;
 }
 int main()
{
   int port; cout << "\nEnter port : "; cin >> port;
   int sock = createServer(port);

  long q, alpha;
  cout<<"\nEnter a prime number, q : "; cin >> q;
  cout<<"Enter primitive root of q, alpha : "; cin >> alpha;

 long Yc;
 recv(sock, &Yc, sizeof(Yc), 0);
 cout<< "\nClient's public key,  Yc = " << Yc <<endl;

 srand(time(NULL));
 long Xs = randInRange(1, q);
 cout<< "\nServer's private key, Xs = " << Xs <<endl;

 long Ys = powermod(alpha, Xs, q);
 send(sock, &Ys, sizeof(Ys), 0);
 cout<< "Server's public key,  Ys = " << Ys <<endl;

 long k = powermod(Yc,Xs,q);
 cout<<"\nSecret Key, k = "<<k<<endl;

 long msg;
 cout<<"\nEnter a message(number) to send : "; cin>>msg;

 long cipher = msg ^ k;
 send(sock, &cipher, sizeof(cipher), 0);
 cout << "Encrypted msg sent to client: " << cipher << endl << endl;
 }
```



```
user@linux-OptiPlex-5090:~/1SI19CS017/diffee-hellman$ g++ server.cpp
user@linux-OptiPlex-5090:~/1SI19CS017/diffee-hellman$ ./a.out

Enter port : 1234

Server Online. Waiting for client....
Connection Established.

Enter a prime number, q : 11
Enter primitive root of q, alpha : 2

Client's public key,  Yc = 9

Server's private key, Xs = 5
Server's public key,  Ys = 10

Secret Key, k = 1

Enter a message(number) to send : anudeep
Encrypted msg sent to client: 1
```



```
user@linux-OptiPlex-5090:~/1SI19CS017/diffee-hellman$ g++ client.cpp
user@linux-OptiPlex-5090:~/1SI19CS017/diffee-hellman$ ./a.out

Enter server's IP address: 127.0.0.1
Enter port : 1234

Client is connected to Server.

Enter a prime number, q : 71
Enter primitive root of q, alpha : 7

Client's private key, Xc = 52
Client's public key,  Yc = 9

Server's public key,  Ys = 10

Secret Key, k = 8

Message received from Server  : 1
Decrpyted message : 9
```

SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMKUR-572103

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CRYPTOGRAPHY AND NETWORK SECURITY LAB (7RCSL01)

Student Name: ANVS Anudeep	**USN: 1SI19CS017**	**Batch No:A1**	**Date:23-01-2023**
Evaluation:			

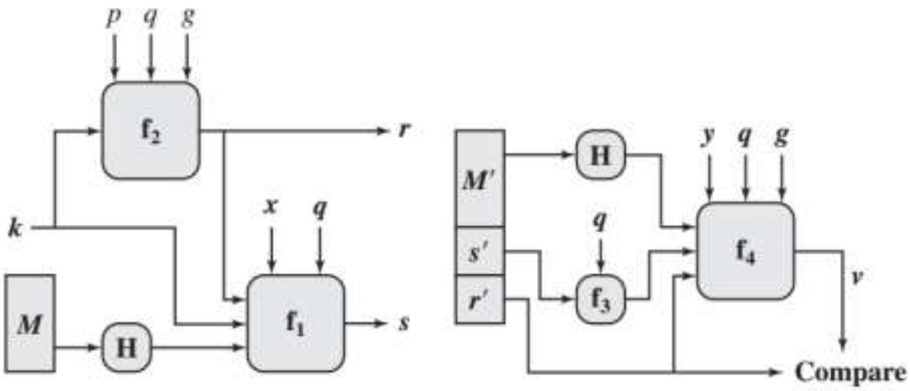
Write Up (10 marks)	**Clarity in concepts (10 marks)**	**Implementation and execution of the algorithms (10 marks)**	**Viva (05 marks)**	**Total (35 marks)**

Sl.No	**Name of the Faculty In-Charge**	**Signature**
1.	Dr AS Poornima	
2.	Ravi V	

Question No: 14

Implement DSS algorithm for signing and verification of messages between two parties (obtain H(M) using simple XOR method of hash computation on M).

Algorithm:



$s = f_1(H(M), k, x, r, q) = (k^{-1}(H(M) + xr)) \bmod q$

$r = f_2(k, p, q, g) = (g^k \bmod p) \bmod q$

$w = f_3(s', q) = (s')^{-1} \bmod q$

$v = f_4(y, q, g, H(M'), w, r')$

$= ((g^{(H(M')w) \bmod q}\, y^{r'w \bmod q}) \bmod p) \bmod$

(a) Signing

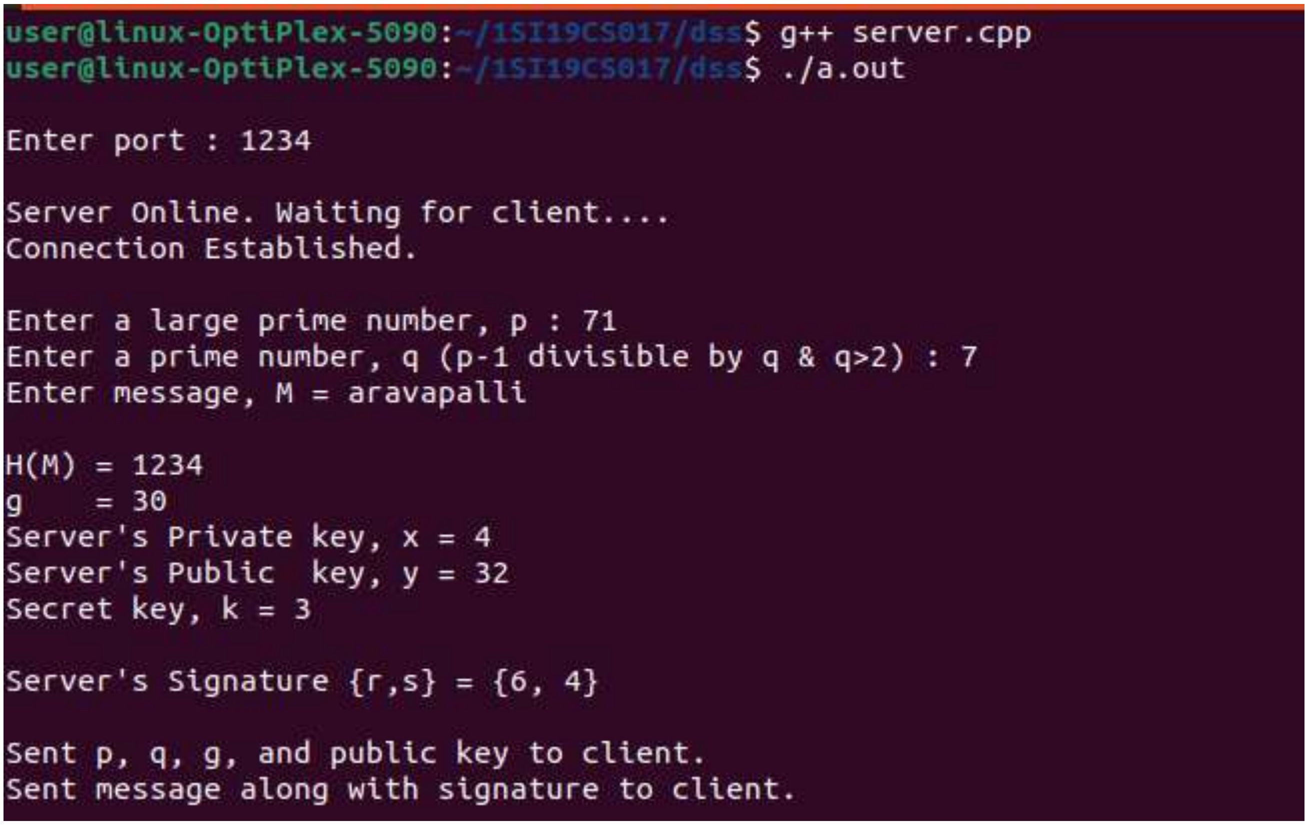
(b) Verifying

Client

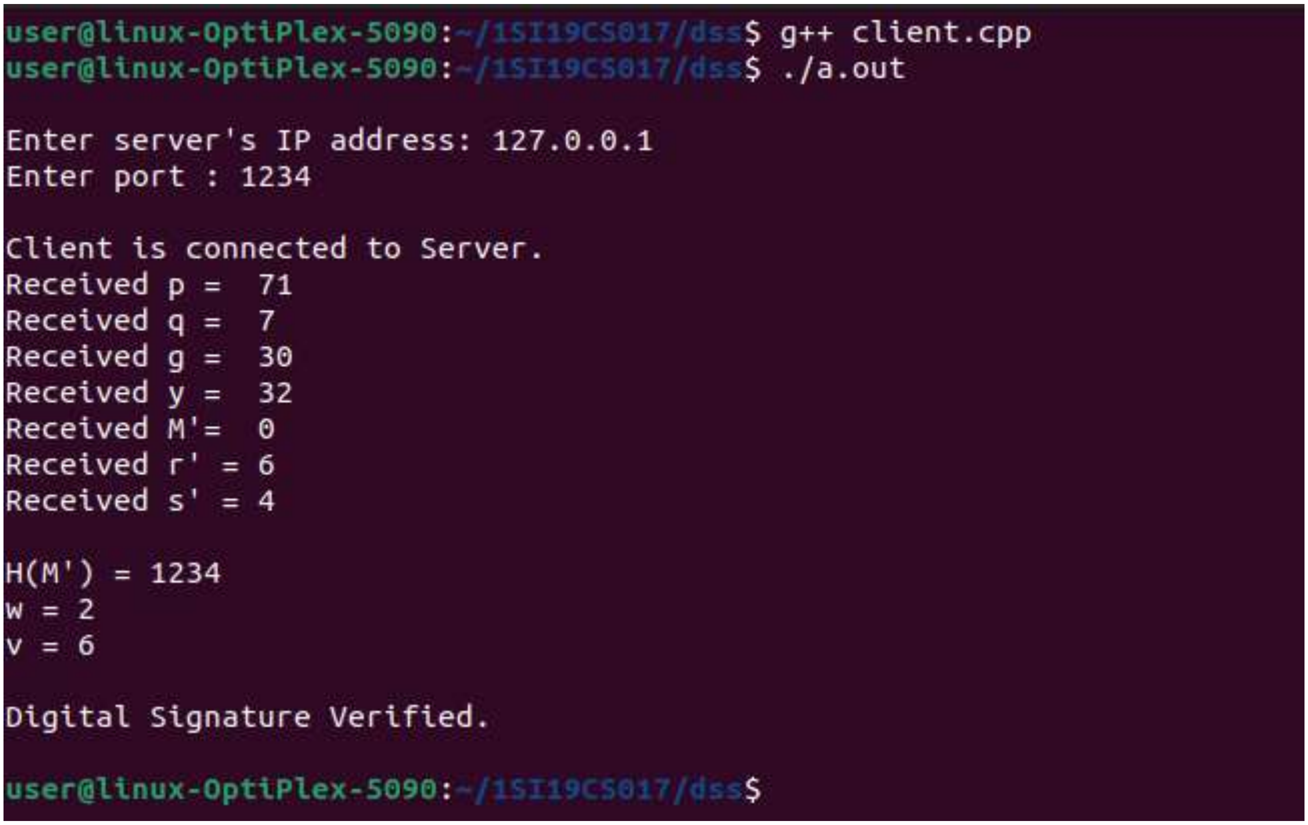
```
# include <bits/stdc++.h>
# include <arpa/inet.h>
using namespace std;

int connectToServer(const char* ip, int port)
{
    int sock = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in addr = {AF_INET, htons(port),inet_addr(ip)};

    if(connect(sock, (struct sockaddr *) &addr, sizeof(addr)) < 0 ){
        cout << "\nRun server program first." << endl; exit(0);
    }else{
        cout << "\nClient is connected to Server." << endl;
    }
    return sock;
}

long mod(long a, long b)
{
    return a >= 0 ? (a%b) : b-(abs(a)%b) ;
}

long powermod(long a, long b, long  c)
{
    long res=1;
    for(int i=0; i<b; i++)
    {
        res = (res * a) % c;
    }
    return res;
}

long findInverse(long R , long D)
{
    int i = 0;
    long N = D;
    long p[100] = {0,1};
    long q[100] = {0} ;

    while(R!=0)
    {
        q[i] = D/R ;
        long oldD = D ;
        D = R ;
        R = oldD%R ;
        if(i>1)
        {
            p[i] = mod(p[i-2] - p[i-1]*q[i-2], N) ;
        }
        i++ ;
    }
    if (i == 1) return 1;
    else        return p[i] = mod(p[i-2] - p[i-1]*q[i-2], N) ;
}

long H(long M)
{
    return (M ^ 1234);
}

int main()
{
    char ip[50]; cout << "\nEnter server's IP address: "; cin >> ip;
    int port;    cout << "Enter port : "; cin >> port;
    int sock = connectToServer(ip, port);

    long p, q;
    long r, s;
    long g, y;
    long M, hashval;
    long w, v;
    srand(time(NULL));

    recv(sock, &p, sizeof(p), 0);
    recv(sock, &q, sizeof(q), 0);
    recv(sock, &g, sizeof(g), 0);
    recv(sock, &y, sizeof(y), 0);
    recv(sock, &M , sizeof(M), 0);
    recv(sock, &r, sizeof(r), 0);
    recv(sock, &s, sizeof(s), 0);

    cout << "Received p =  " << p << endl;
    cout << "Received q =  " << q << endl;
    cout << "Received g =  " << g << endl;
    cout << "Received y =  " << y << endl;
    cout << "Received M'=  " << M << endl;
    cout << "Received r' = " << r << endl;
    cout << "Received s' = " << s << endl;

    hashval = H(M) ;
    cout << "\nH(M') = " << hashval << endl;

    w = findInverse(s,q) % q;  cout << "w = " << w << endl;
    long u1 = (hashval * w) % q;
    long u2 = (r * w) % q;
    v = ((powermod(g,u1,p)*powermod(y,u2,p)) %p) %q;  cout<<"v = "<<v<<endl;
    if(v == r) cout<<"\nDigital Signature Verified. " << endl << endl;
    else   cout<<"\nDigital Signature is invalid !!!" << endl << endl;
}
```

Server

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

int createServer(int port)
{
    int sersock = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in addr = {AF_INET, htons(port), INADDR_ANY};

    bind(sersock, (struct sockaddr *) &addr, sizeof(addr));
    cout << "\nServer Online. Waiting for client...." << endl;

    listen(sersock, 5);
    int sock = accept(sersock, NULL, NULL);
    cout << "Connection Established." << endl;
    return sock;
}

long randInRange(long low, long high)
{
    return rand()%(high-(low+1)) + (low+1) ;
}

long mod(long a, long b)
{
return a >= 0 ? (a%b) : b-(abs(a)%b) ;
}

long powermod(long a, long b, long  c)
{
    long res=1;
    for(int i=0; i<b; i++)
    {
        res = (res * a) % c;
    }
    return res;
}

long findInverse(long R , long D)
{
    int i = 0;
    long N = D;
    long p[100] = {0,1};
    long q[100] = {0} ;

    while(R!=0)
    {
        q[i] = D/R ;
        long oldD = D ;
        D = R ;
        R = oldD%R ;
        if(i>1)
        {
            p[i] = mod(p[i-2] - p[i-1]*q[i-2], N) ;
        }
        i++ ;
    }
    if (i == 1) return 1;
    else        return p[i] = mod(p[i-2] - p[i-1]*q[i-2], N) ;
}

long H(long M)
{
return (M ^ 1234);
}

int main()
{
    int port;  cout << "\nEnter port : "; cin >> port;
    int sock = createServer(port);

    long p, q;
    long r, s;
    long k, x, y, g;
    long M, hashval;
    srand(time(NULL));

    cout << "\nEnter a large prime number, p : ";   cin >> p;
    cout << "Enter a prime number, q (p-1 divisible by q & q>2) : "; cin >> q;
    if( (p-1)%q != 0 || q <3) { cout << "\nInvalid input\n"; exit(-1); }

    cout<<"Enter message, M = "; cin >> M;

    hashval = H(M);
    cout << "\nH(M) = " << hashval << endl;

    long h;
    do{
        h = randInRange(1, p-1);
        g = powermod(h,(p-1)/q, p);
    } while(g<=1);
    cout << "g    = " << g;
    x = randInRange(1, q);  cout << "\nServer's Private key, x = " << x;
    y = powermod(g, x, p);  cout << "\nServer's Public  key, y = " << y;
    k = randInRange(1, q);  cout << "\nSecret key, k = " << k << endl;


    r = powermod(g, k, p) % q;
    s = (findInverse(k,q) * (hashval + x*r )) % q;
    cout << "\nServer's Signature {r,s} = {" << r << ", " << s << "}" << endl;

    send(sock, &p, sizeof(p), 0);
    send(sock, &q, sizeof(q), 0);
    send(sock, &g, sizeof(g), 0);
    send(sock, &y, sizeof(y), 0);
    send(sock, &M , sizeof(M), 0);
    send(sock, &r, sizeof(r), 0);
    send(sock, &s, sizeof(s), 0);

    cout << "\nSent p, q, g, and public key to client.";
    cout <<"\nSent message along with signature to client." << endl << endl;
}
```



```
user@linux-OptiPlex-5090:~/1SI19CS017/dss$ g++ server.cpp
user@linux-OptiPlex-5090:~/1SI19CS017/dss$ ./a.out

Enter port : 1234

Server Online. Waiting for client....
Connection Established.

Enter a large prime number, p : 71
Enter a prime number, q (p-1 divisible by q & q>2) : 7
Enter message, M = aravapalli

H(M) = 1234
g    = 30
Server's Private key, x = 4
Server's Public  key, y = 32
Secret key, k = 3

Server's Signature {r,s} = {6, 4}

Sent p, q, g, and public key to client.
Sent message along with signature to client.
```



```
user@linux-OptiPlex-5090:~/1SI19CS017/dss$ g++ client.cpp
user@linux-OptiPlex-5090:~/1SI19CS017/dss$ ./a.out

Enter server's IP address: 127.0.0.1
Enter port : 1234

Client is connected to Server.
Received p =  71
Received q =  7
Received g =  30
Received y =  32
Received M'=  0
Received r' = 6
Received s' = 4

H(M') = 1234
w = 2
v = 6

Digital Signature Verified.

user@linux-OptiPlex-5090:~/1SI19CS017/dss$
```