01. Write a program to perform the following using Playfair cipher technique
(i) Encrypt a given message M with different keys {k1,k2,…,kn}. Print key and cipher text pair
(ii) Decrypt the cipher texts obtained in (i) to get back M

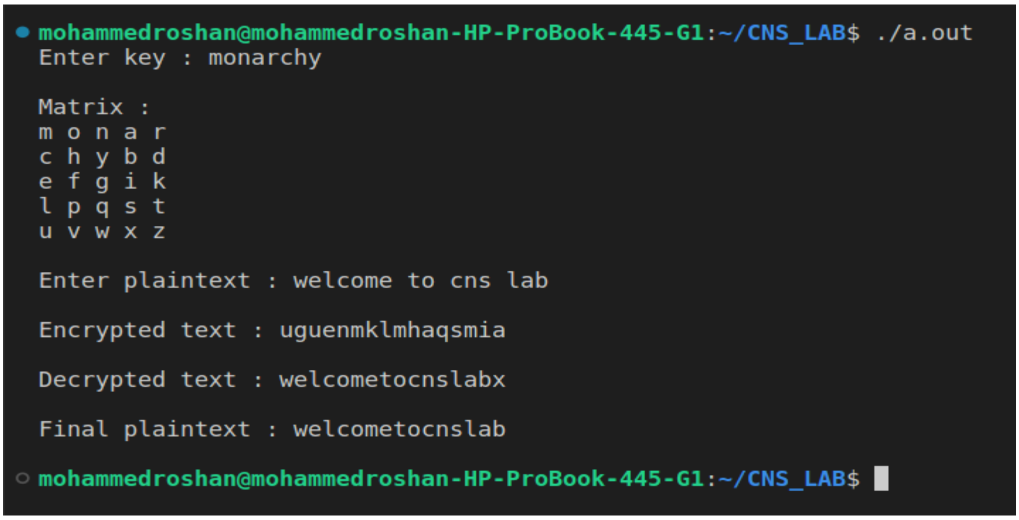
Code :

```
#include<bits/stdc++.h>
using namespace std;
/*function to find mod of number ex:- 2 % 5 = 0 ,  -1 % 5 = 4  ,  5 % 5 = 0  */

int mod(int n){
   if(n>=0) return n%5;
   return n%5+5;
}
/* function for both encrypting and decrypting the given text */

string encrpt_decrypt(map<char,int> m1,map<int,char> m2,string text,int n,int op){
   int temp1,temp2,frow,fcol,srow,scol,temp;
   string cipher="";
   for(int i=0;i<n;i+=2){
      temp1=m1[text[i]];
      temp2=m1[text[i+1]];
      frow=temp1/5;
      fcol=temp1%5;
      srow=temp2/5;
      scol=temp2%5;
      if(frow==srow){
         fcol=mod(fcol+op);
         scol=mod(scol+op);
      }
      else if(fcol==scol){
         frow=mod(frow+op);
         srow=mod(srow+op);
      }
      else{
         swap(fcol,scol);
      }
      cipher=cipher+m2[frow*5+fcol]+m2[srow*5+scol];
   }
return cipher;
}
int main(){
   string key;
   cout<<"Enter key : ";
   cin>>key;
   int n=key.length();
   int i,c=0;

   map<char,int> m1;
```

```
map<int,char> m2;

for(i=0;i<n;i++){
   if(isupper(key[i]))
   key[i]=key[i]+32;

   if(key[i]=='j')
   key[i]='i';

   if(!m1.count(key[i]))
   {
      m1[key[i]]=c;
      m2[c]=key[i];
      c=c+1;
   }
}
for(char ch='a';ch<='z';ch++){
   if(ch!='j' and !m1.count(ch))
   {
      m1[ch]=c;
      m2[c]=ch;
      c=c+1;
   }
}
/* To print matrix formed from the given key */
cout<<endl<<"Matrix : ";
for(auto i:m2){
   if(i.first%5==0)
   cout<<endl;

   cout<<m2[i.first]<<" ";
}
string plaintext,processed_plaintext="";

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

n=plaintext.length();

for(i=0;i<n;i++){
   if(isalpha(plaintext[i])){

      if(isupper(plaintext[i]))
      plaintext[i]=plaintext[i]+32;

      if(plaintext[i]==processed_plaintext.back())
```

```
            processed_plaintext+='x';

            processed_plaintext+=plaintext[i];
        }
    }

    if(processed_plaintext.length()%2==1)
    processed_plaintext+='x';

    n=processed_plaintext.length();

    string enc_text=encrpt_decrypt(m1,m2,processed_plaintext,n,1);
    cout<<endl<<"Encrypted text : "<<enc_text<<endl;

    string dec_text=encrpt_decrypt(m1,m2,enc_text,n,-1);
    cout<<endl<<"Decrypted text : "<<dec_text<<endl;

    plaintext="";
    n=dec_text.length();
    plaintext=dec_text[0];

    for(i=1;i<n-1;i++){
        if(!(dec_text[i]=='x' and dec_text[i-1]==dec_text[i+1]))
        plaintext+=dec_text[i];
    }

    if(dec_text[n-1]!='x')
    plaintext+=dec_text[i];

    cout<<endl<<"Final plaintext : "<<plaintext<<endl<<endl;
    return 0;
}
```

Output :



```
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/CNS_LAB$ ./a.out
Enter key : monarchy

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

Enter plaintext : welcome to cns lab

Encrypted text : uguenmklmhaqsmia

Decrypted text : welcometocnslabx

Final plaintext : welcometocnslab

mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/CNS_LAB$
```

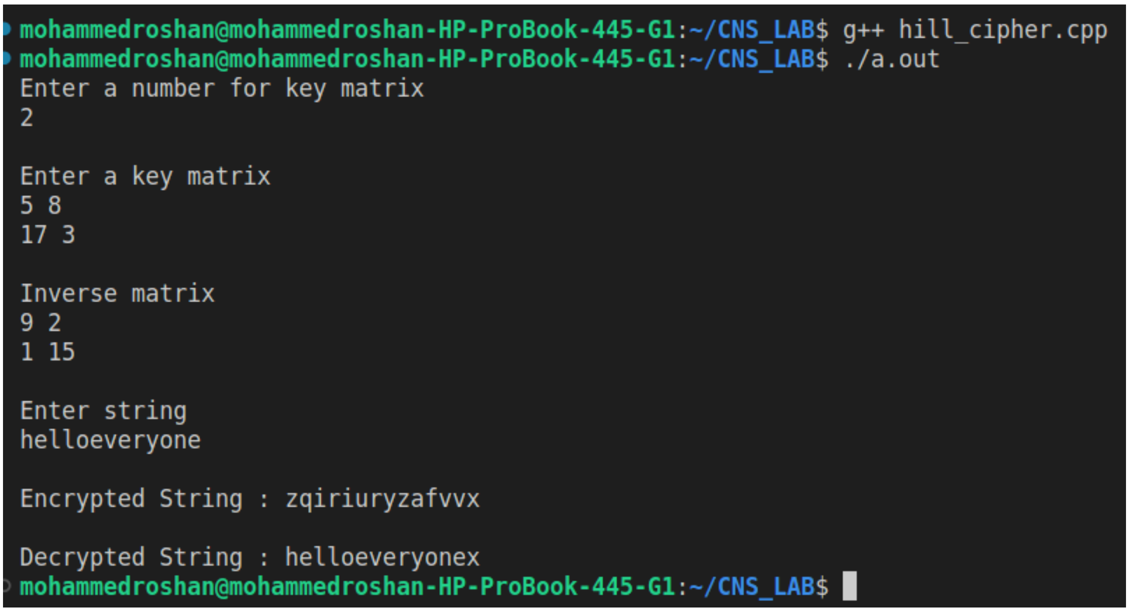
2. Write a program to perform the following using Hill cipher:
(i) Encrypt a message M with a given key matrix of size 2X2 and 3X3
(ii) Decrypt the cipher text obtained in (i) by computing inverse of the respective key matrix.

Code :

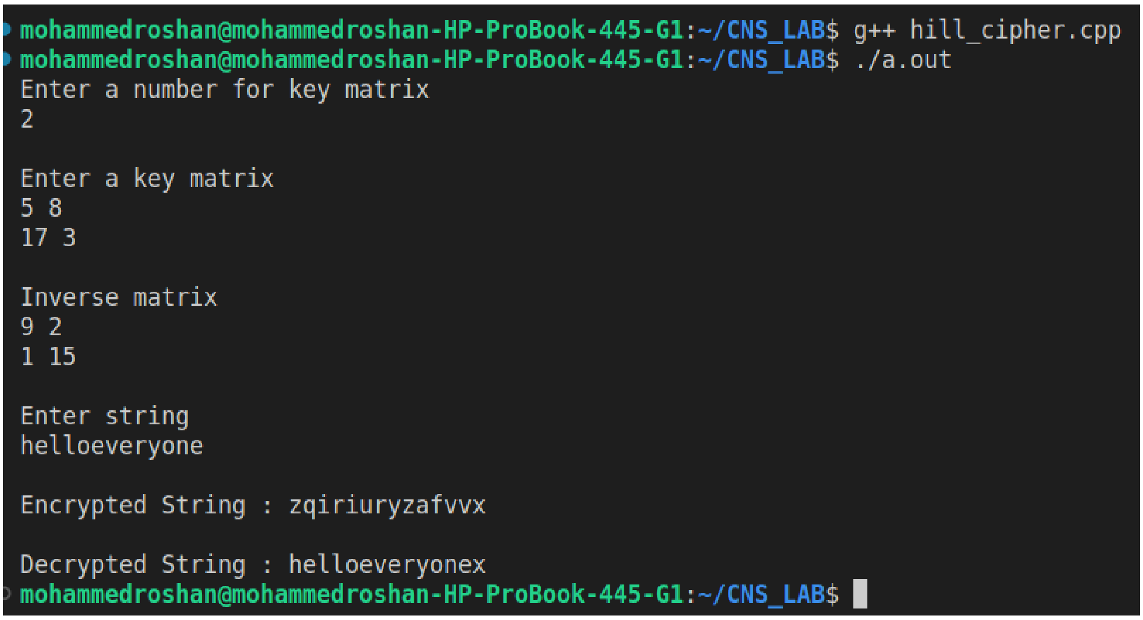
```
#include<bits/stdc++.h>
using namespace std;

/* function to mod of given number ex:- 61 % 26 = 9 , -121 % 26 = */
int modulo(int n){
        if(n>=0) return n%26;
        return (n%26)+26;
}

/* function to find gcd of given two numbers */
int gcd(int a,int b){
        if(b==0) return a;
        return gcd(b,a%b);
}

/* function to find multiplicative inverse using extented euclidean algorithm */
int gcd1(int a,int b,int p,int q){
        if(b==0) return modulo(p);
        return gcd1(b,a%b,q,p-(a/b)*q);
}

/* function to find determinant of 2x2 or 3x3 matrix */
int det(int a[3][3],int n){
        if(n==2) return modulo(a[0][0]*a[1][1]-a[0][1]*a[1][0]);
        return modulo(a[0][0]*(a[1][1]*a[2][2]-a[1][2]*a[2][1])

-a[0][1]*(a[1][0]*a[2][2]-a[1][2]*a[2][0])+a[0][2]*(a[1][0]*a[2][1]-a[1][1]*a[2][0]));
}

/* function to find inverse of 2x2 or 3x3 matrix */
void inverse(int a[3][3],int inv[3][3],int n,int mul_inv){
        if(n==2){
                inv[0][0]=modulo(a[1][1]*mul_inv);
                inv[1][1]=modulo(a[0][0]*mul_inv);
                inv[0][1]=modulo(-a[0][1]*mul_inv);
                inv[1][0]=modulo(-a[1][0]*mul_inv);
        }
        else{

                int i,j,temp;
                inv[0][0]=modulo((a[1][1]*a[2][2]-a[1][2]*a[2][1])*mul_inv);
                inv[0][1]=modulo(-(a[1][0]*a[2][2]-a[1][2]*a[2][0])*mul_inv);
                inv[0][2]=modulo((a[1][0]*a[2][1]-a[1][1]*a[2][0])*mul_inv);
                inv[1][0]=modulo(-(a[0][1]*a[2][2]-a[0][2]*a[2][1])*mul_inv);
                inv[1][1]=modulo((a[0][0]*a[2][2]-a[0][2]*a[2][0])*mul_inv);
                inv[1][2]=modulo(-(a[0][0]*a[2][1]-a[0][1]*a[2][0])*mul_inv);
                inv[2][0]=modulo((a[0][1]*a[1][2]-a[0][2]*a[1][1])*mul_inv);
                inv[2][1]=modulo(-(a[0][0]*a[1][2]-a[0][2]*a[1][0])*mul_inv);
                inv[2][2]=modulo((a[0][0]*a[1][1]-a[0][1]*a[1][0])*mul_inv);

                /* code snippet to tranpose the given matrix */
                for(i=0;i<3;i++)
                for(j=0;j<i;j++){
                temp=inv[i][j];
                inv[i][j]=inv[j][i];
                inv[j][i]=temp;
                }
        }
}

/* function to find matrix multiplication of given two matrices */
void mat_mul(int mat[100][3],int res[100][3],int a[3][3],int row,int n){
        int i,j,k;
        for(i=0;i<row;i++){
                for(j=0;j<n;j++){
                res[i][j]=0;
                for(k=0;k<n;k++)
                res[i][j]+=mat[i][k]*a[k][j];
                res[i][j]=modulo(res[i][j]);
                }
        }
}

/* function to print elements of matrix */
void display(int mat[100][3],int row,int n){
        for(int i=0;i<row;i++){
                for(int j=0;j<n;j++){
                        cout<<char(mat[i][j]+97);
                }
        }
        cout<<endl;
}

int main(){

int n,i,j;
cout<<"Enter a number for key matrix\n";
cin>>n;
```

```
int a[3][3],inv[3][3];

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

int dt=det(a,n);

/* mutiplicative inverse only exists , if and only if 26 and determinant value of key matrix are relatvely prime */

if(gcd(dt,26)==1){

        int mul_inv=gcd1(26,dt,0,1);

        inverse(a,inv,n,mul_inv);

        cout<<endl<<"Inverse matrix\n";
        for(i=0;i<n;i++){
                for(j=0;j<n;j++){
                        cout<<inv[i][j]<<" ";
                }
                cout<<endl;
        }

        string s;
        cout<<endl<<"Enter string\n";
        cin>>s;

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

        int len=s.length();

        int row=len/n;
        int mat[100][3],res[100][3],dec[100][3];
        int y=0;

        /* code to fill plaintext into the matrix row wise */
        for(i=0;i<len;i++){
                mat[i/n][i%n]=s[i]-97;
        }
        mat_mul(mat,res,a,row,n);
        cout<<endl<<"Encrypted String : ";
        display(res,row,n);

        cout<<endl<<"Decrypted String : ";
        mat_mul(res,dec,inv,row,n);
        display(dec,row,n);
}
else {
        cout<<"Inverse of given key matrix doesn't exist"<<endl;
}
return 0;
}
```

Output :



```
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/CNS_LAB$ g++ hill_cipher.cpp
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/CNS_LAB$ ./a.out
Enter a number for key matrix
2

Enter a key matrix
5 8
17 3

Inverse matrix
9 2
1 15

Enter string
helloeveryone

Encrypted String : zqiriuryzafvvx

Decrypted String : helloeveryonex
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/CNS_LAB$
```



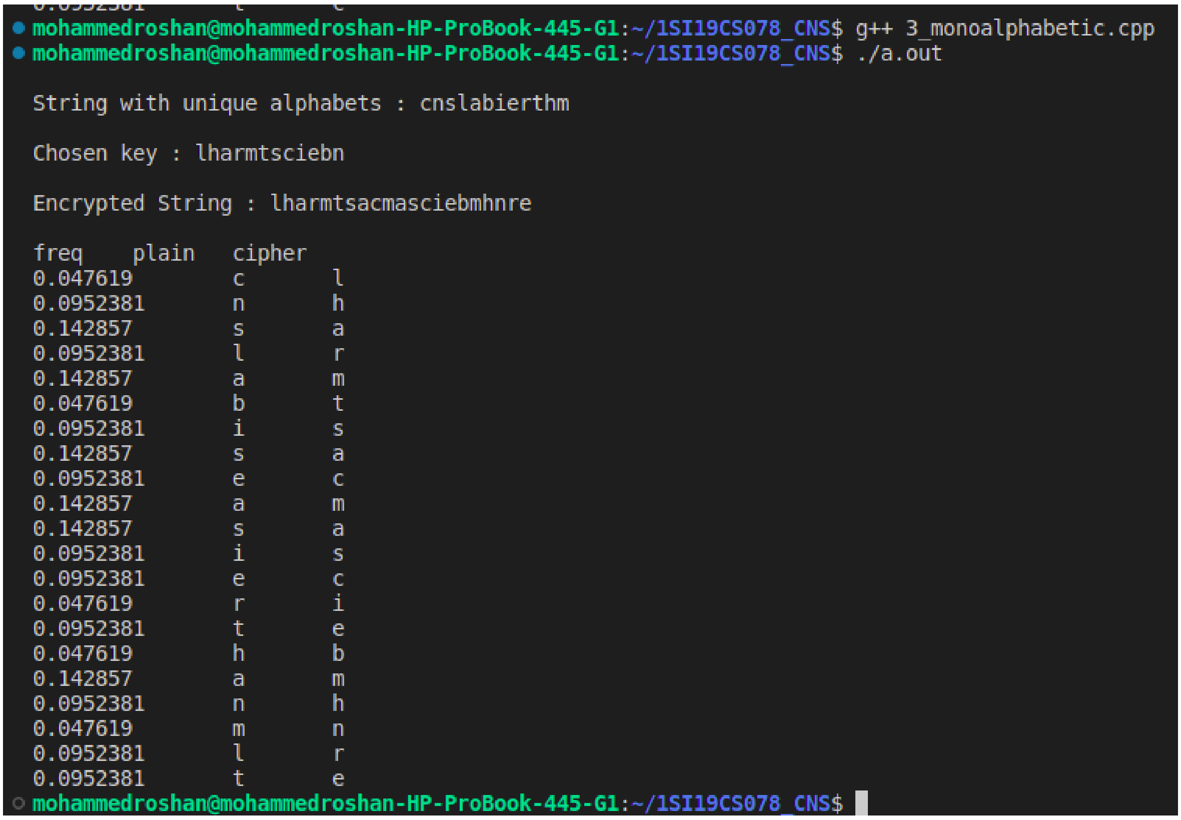
```
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/CNS_LAB$ g++ hill_cipher.cpp
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/CNS_LAB$ ./a.out
Enter a number for key matrix
2

Enter a key matrix
5 8
17 3

Inverse matrix
9 2
1 15

Enter string
helloeveryone

Encrypted String : zqiriuryzafvvx

Decrypted String : helloeveryonex
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/CNS_LAB$
```

3. 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. Encrypt the characters of plaintext.txt and store the corresponding ciphertext characters in ciphertext.txt
iii. Compute the frequency of occurrence of each alphabet in both plaintext.txt and ciphertext.txt and tabulate the results

Code :

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    srand(time(NULL));
    int i,n;

    ifstream fin;
    string plaintext;
    fin.open("plaintext.txt");
    fin>>plaintext;
    fin.close();

    set<char> s;
    string org="";
    n=plaintext.length();

    for(i=0;i<n;i++){
        if(!s.count(plaintext[i])){
            org+=plaintext[i];
            s.insert(plaintext[i]);
        }
    }
    string dup=org;
    n=org.length();
    for(i=0;i<n;i++){
        int c=rand()%n;
        char ch=dup[i];
        dup[i]=dup[c];
        dup[c]=ch;
    }

    map<char,char> m;
    for(i=0;i<n;i++){
        m[org[i]]=dup[i];
    }
    string cipher="";
    map<char,int> freq;
    n=plaintext.length();
    for(i=0;i<n;i++){
        cipher+=m[plaintext[i]];
        freq[plaintext[i]]+=1;
    }

    cout<<endl<<"String with unique alphabets : "<<org<<endl<<endl;
    cout<<"Chosen key : "<<dup<<endl<<endl;
    cout<<"Encrypted String : "<<cipher<<endl<<endl;

    ofstream fout;
    fout.open("ciphertext.txt");
    fout<<cipher;
    fout.close();

    cout<<"freq\tplain\tcipher"<<endl;
    for(i=0;i<n;i++){
        cout<<float(freq[plaintext[i]])/n<<"\t"<<plaintext[i]<<"\t"<<cipher[i]<<endl;
    }
  return 0;
}
```

Output :



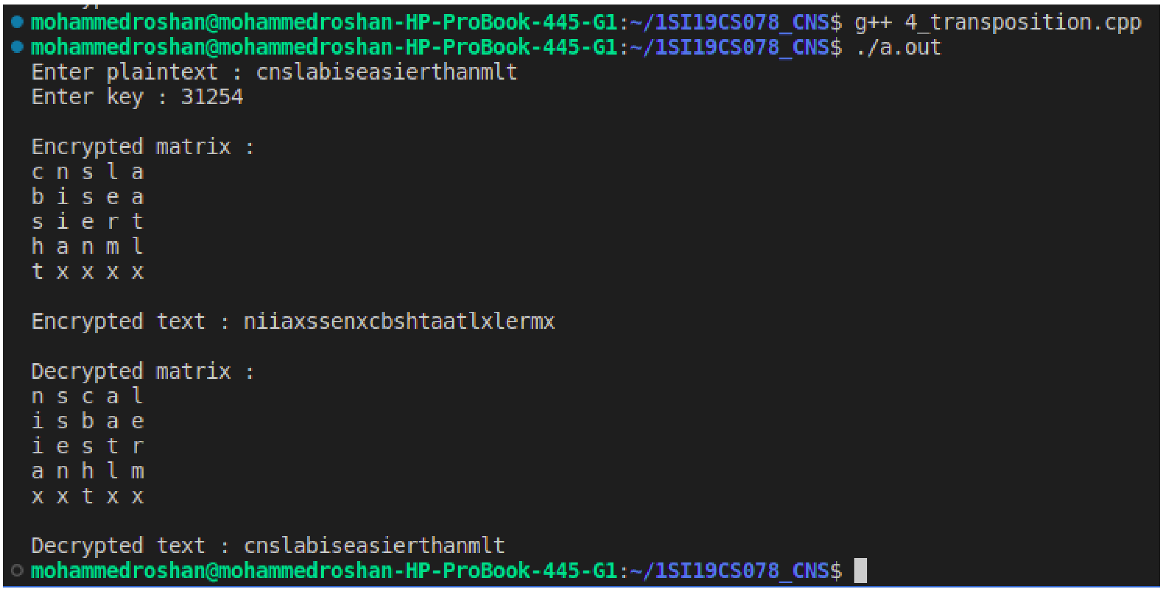
```
● mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ g++ 3_monoalphabetic.cpp
● mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ ./a.out

  String with unique alphabets : cnslabierthm

  Chosen key : lharmtsciebn

  Encrypted String : lharmtsacmasciebmhnre

  freq    plain   cipher
  0.047619        c       l
  0.0952381       n       h
  0.142857        s       a
  0.0952381       l       r
  0.142857        a       m
  0.047619        b       t
  0.0952381       i       s
  0.142857        s       a
  0.0952381       e       c
  0.142857        a       m
  0.142857        s       a
  0.0952381       i       s
  0.0952381       e       c
  0.047619        r       i
  0.0952381       t       e
  0.047619        h       b
  0.142857        a       m
  0.0952381       n       h
  0.047619        m       n
  0.0952381       l       r
  0.0952381       t       e
○ mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ 
```

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

Code :

```
#include<bits/stdc++.h>
using namespace std;
void display(char mat[10][10],int rows,int cols){
    for(int i=0;i<rows;i++)
    {
        for(int j=0;j<cols;j++)
        cout<<mat[i][j]<<" ";
        cout<<endl;
    }
}
string encrpyt(string text,string key,int n,int cols){
    int i,j,cur_col;
    int rows=n/cols;
    char mat[10][10];

    for(i=0;i<n;i++)
        mat[i/cols][i%cols]=text[i];

    display(mat,rows,cols);

    string cipher="";
    for(i=0;i<cols;i++){
        cur_col=key.find(i+'1');
        for(j=0;j<rows;j++){
            cipher+=mat[j][cur_col];
        }
    }
    return cipher;
}
string decrypt(string text,string key,int n,int cols){
    int i,j;
    int rows=n/cols;
    char mat[10][10];

    for(i=0;i<n;i++)
        mat[i%rows][i/rows]=text[i];

    display(mat,rows,cols);

    string cipher="";
    for(i=0;i<rows;i++){
        for(j=0;j<cols;j++){
            cipher+=mat[i][key[j]-'1'];
        }
    }
    return cipher;
}
int main(){
    string key,plaintext;
    cout<<"Enter plaintext : ";
    cin>>plaintext;
    cout<<"Enter key : ";
    cin>>key;
    int cols=key.length();
    while(plaintext.length()%cols!=0)
    plaintext+="x";

    int n=plaintext.length();
    cout<<endl<<"Encrypted matrix : "<<endl;
    string enc_text=encrpyt(plaintext,key,n,cols);
    cout<<endl<<"Encrypted text : "<<enc_text<<endl;

    cout<<endl<<"Decrypted matrix : "<<endl;
    string dec_text=decrypt(enc_text,key,n,cols);

    while(dec_text.back()=='x') dec_text.pop_back();

    cout<<endl<<"Decrypted text : "<<dec_text<<endl;
    return 0;
}
```

Output :



```
● mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ g++ 4_transposition.cpp
● mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ ./a.out
  Enter plaintext : cnslabiseasierthanmlt
  Enter key : 31254

  Encrypted matrix :
  c n s l a
  b i s e a
  s i e r t
  h a n m l
  t x x x x

  Encrypted text : niiaxssenxcbshtaatlxlermx

  Decrypted matrix :
  n s c a l
  i s b a e
  i e s t r
  a n h l m
  x x t x x

  Decrypted text : cnslabiseasierthanmlt
○ mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ 
```

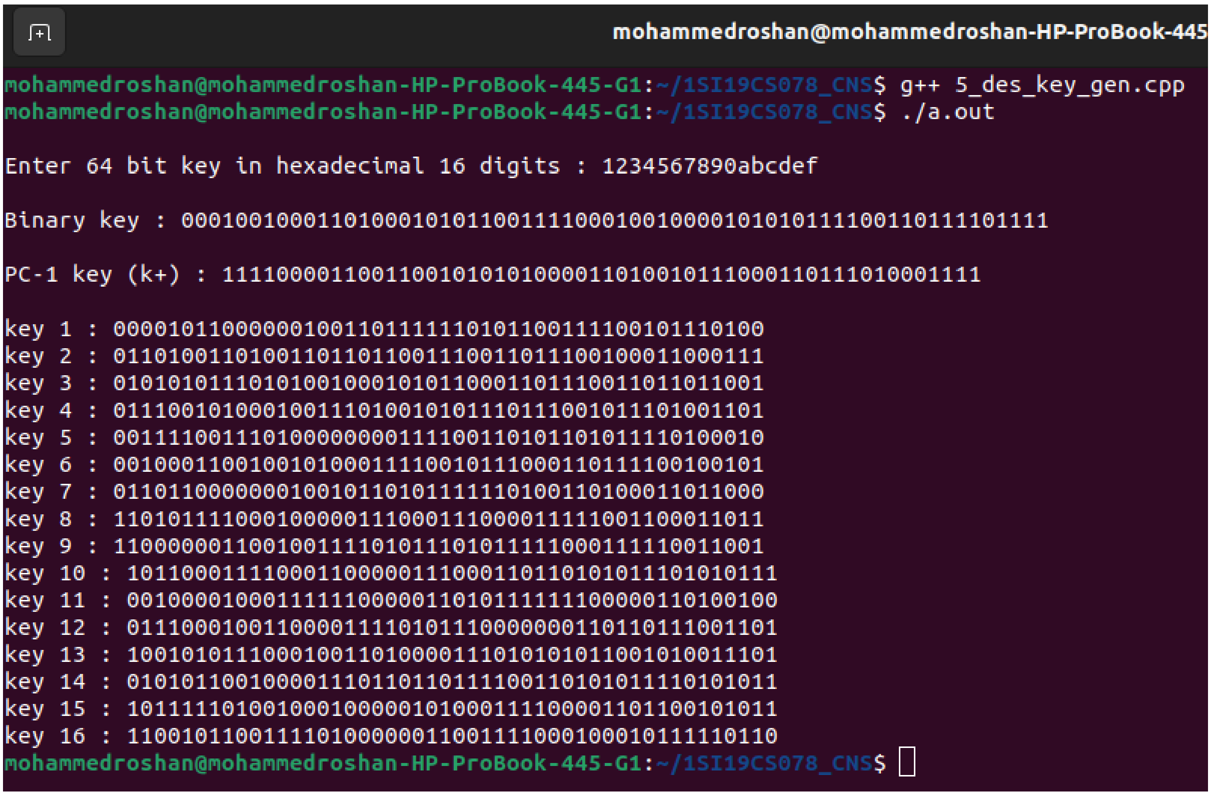
5. Generate and print 48-bit keys for all sixteen roundsof DES algorithm, given a 64-bit initial key.

Code :

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

int permute_one[]= {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 permute_two[] = {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 leftshift(string text,int n){
          return text.substr(n,text.length()-n)+text.substr(0,n);
}

string firstPermute(string key){
          string res="";
          for(int i=0;i<56;i++)
                    res+=key[permute_one[i]-1];
          return res;
}
string secondPermute(string key){
          string res="";
          for(int i=0;i<48;i++)
                    res+=key[permute_two[i]-1];
          return res;
}
void gen_keys(string left,string right){
          string key;
          for(int i=0;i<16;i++){
                    left=leftshift(left,leftshiftTable[i]);
                    right=leftshift(right,leftshiftTable[i]);
```

```
                key=secondPermute(left+right);

                cout<<"key "<<i+1<<" : "<<key<<endl;
        }
}
int main(){
        unsigned long long key;
        cout<<endl<<"Enter 64 bit key in hexadecimal 16 digits : ";
        cin>>hex>>key;
        string binarykey=bitset<64>(key).to_string();
        cout<<endl<<"Binary key : "<<binarykey<<endl;

        binarykey=firstPermute(binarykey);

        cout<<endl<<"PC-1 key (k+) : "<<binarykey<<endl<<endl;
        gen_keys(binarykey.substr(0,28),binarykey.substr(28,28));


        return 0;
}
```

<u>Output</u> :



```
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ g++ 5_des_key_gen.cpp
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ ./a.out

Enter 64 bit key in hexadecimal 16 digits : 1234567890abcdef

Binary key : 0001001000110100010101100111100010010000101010111100110111101111

PC-1 key (k+) : 11110000110011001010101000011010010111000110111010001111

key 1 : 000010110000000100110111111010110011110010111010
key 2 : 011010011010011011011001110011011100100011000111
key 3 : 010101011101010010001010110001101110011011011001
key 4 : 011100101000100111010010101110111001011101001101
key 5 : 001111001110100000000011110011010110101111010001
key 6 : 001000110010010100011110010111000110111100100101
key 7 : 011011000000010010110101111110100110100011011000
key 8 : 110101111000100000111000111000011111001100011011
key 9 : 110000001100100111101011101011111000111110011001
key 10 : 101100011110001100000111000110110101011101010111
key 11 : 001000010001111110000011010111111110000011010010
key 12 : 011100010011000011110101110000000110110111001101
key 13 : 100101011100010011010000111010101011001010011101
key 14 : 010101100100001110110110111100110101011110101011
key 15 : 101111101001000100000101000111100001101100101011
key 16 : 110010110011110100000011001111000100010111110110
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ 
```

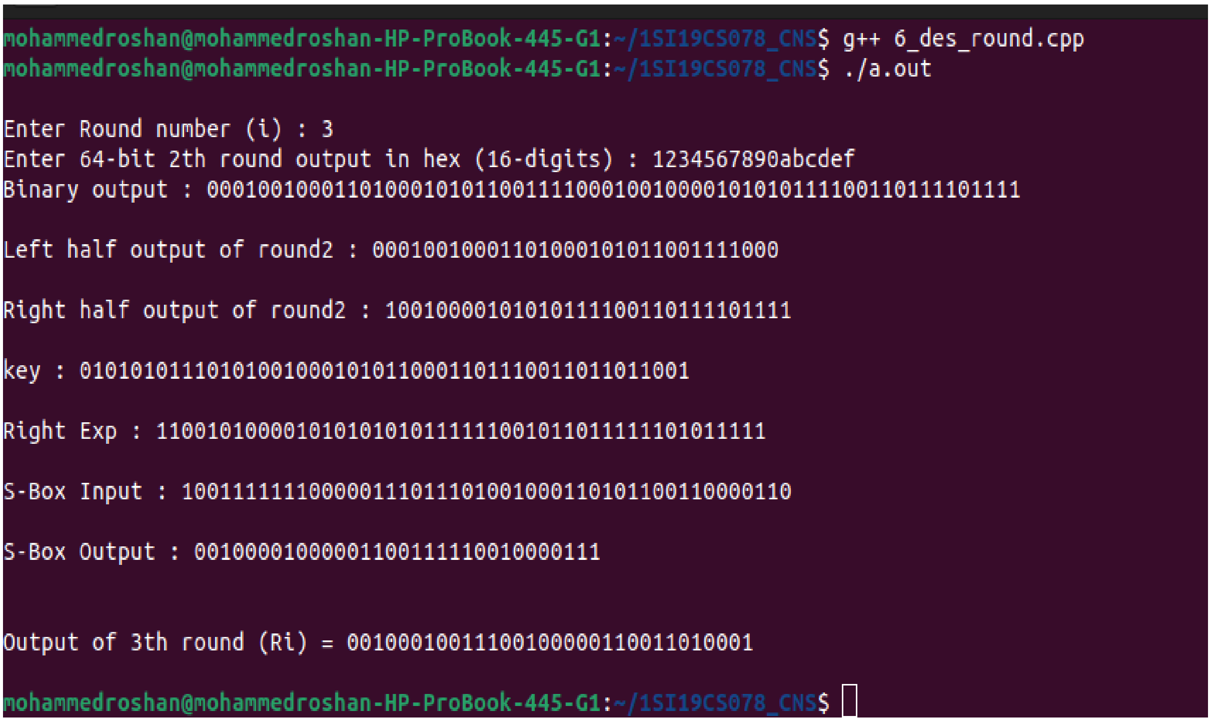
6. i. Given 64-bit output of (i-1)th round of DES, 48-bit ith round key Ki and E table, find the 48-bit input for S-box.
ii. Given 48-bit input to S-box and permutation table P, find the 32-bit output Ri of ith round of DES algorithm.

Code :

```
#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}
};

string permute(string key,int arr[],int n){
        string res="";
        for(int i=0;i<n;i++)
                res+=key[arr[i]-1];
        return res;
}
string xor_(string str1,string str2){
        string res="";
        for(int i=0;i<str1.length();i++){
                if (str1[i]==str2[i]) res+="0";
                else res+="1";
        }
        return res;
}
string s_box_substiution(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;
}
int main(){

        int E[] = {
                        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 };
  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 };
        int r;
        cout << "\nEnter Round number (i) : ";
        cin >> r;
        ifstream fin;
        fin.open("keygen.txt");
        string key;
        for(int j=0;j<r;j++)
        fin>>key;
        if(key.length()==0)
        {
                cout<<"Key not found\n";
                exit(0);
        }
        unsigned long long hexinput;
        cout << "Enter 64-bit "<<r-1<<"th round output in hex (16-digits) : " ;
        cin >> hex >> hexinput;
        string input = bitset<64>(hexinput).to_string();
        cout<<"Binary output : "<<input<<endl;

        string left=input.substr(0,32);
        string right=input.substr(32,32);



        cout<<endl<<"Left half output of round"<<r-1<<" : "<<left<<endl;
        cout<<endl<<"Right half output of round"<<r-1<<" : "<<right<<endl;
        cout<<endl<<"key : "<<key<<endl;

        string right_exp=permute(right,E,48);
        cout<<endl<<"Right Exp : "<<right_exp<<endl;
        string s_box_input=xor_(right_exp,key);

        cout<<endl<<"S-Box Input : "<<s_box_input<<endl;
        /* Till here 6th program's first part ends.For calculating first part of 6th program 8 s-boxes are
not needed*/

        string s_box_output=s_box_substiution(s_box_input);
        cout<<endl<<"S-Box Output : "<<s_box_output<<endl;

        string per=permute(s_box_output,permTable,32);
        string updated_right=xor_(left,per);

        cout <<endl<< "\nOutput of "<<r<<"th round (Ri) = " << updated_right<< endl << endl;

        return 0;
```

}

Output :



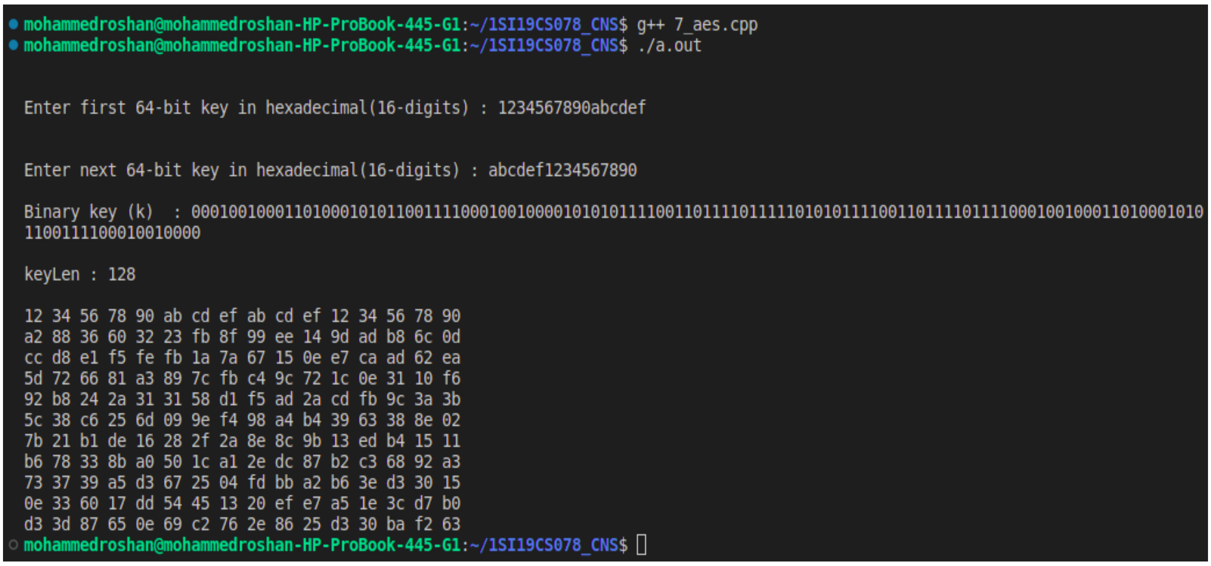
```
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ g++ 6_des_round.cpp
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ ./a.out

Enter Round number (i) : 3
Enter 64-bit 2th round output in hex (16-digits) : 1234567890abcdef
Binary output : 0001001000110100010101100111100010010000101010111100110111101111

Left half output of round2 : 00010010001101000101011001111000

Right half output of round2 : 10010000101010111100110111101111

key : 010101011101010010001010110001101110011011011001

Right Exp : 110010100001010101010111111001011011111101011111

S-Box Input : 100111111100000111011101001000110101100110000110

S-Box Output : 00100001000000110011111001000111


Output of 3th round (Ri) = 00100010011100100000110011010001

mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ 
```

7. Consider the 128 bits initial key and expand it to 10 different keys each of size 128 bits using AES key expansion technique.

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 <<endl<< "\nEnter first 64-bit key in hexadecimal(16-digits) : " ;
   cin >> hex >> hexkey1;
   cout <<endl<< "\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 <<endl<< "Binary key (k) \t: " << key << endl;
   cout <<endl<< "keyLen : " << key.length() << endl<<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/4-1]).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++){
      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;
}
```

Output :



```
● mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ g++ 7_aes.cpp
● mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ ./a.out


  Enter first 64-bit key in hexadecimal(16-digits) : 1234567890abcdef


  Enter next 64-bit key in hexadecimal(16-digits) : abcdef1234567890

  Binary key (k)  : 0001001000110100010101100111100010010000101010111100110111101111101010111100110111101111000100100011010001010
  1100111100010010000

  keyLen : 128

  12 34 56 78 90 ab cd ef ab cd ef 12 34 56 78 90
  a2 88 36 60 32 23 fb 8f 99 ee 14 9d ad b8 6c 0d
  cc d8 e1 f5 fe fb 1a 7a 67 15 0e e7 ca ad 62 ea
  5d 72 66 81 a3 89 7c fb c4 9c 72 1c 0e 31 10 f6
  92 b8 24 2a 31 31 58 d1 f5 ad 2a cd fb 9c 3a 3b
  5c 38 c6 25 6d 09 9e f4 98 a4 b4 39 63 38 8e 02
  7b 21 b1 de 16 28 2f 2a 8e 8c 9b 13 ed b4 15 11
  b6 78 33 8b a0 50 1c a1 2e dc 87 b2 c3 68 92 a3
  73 37 39 a5 d3 67 25 04 fd bb a2 b6 3e d3 30 15
  0e 33 60 17 dd 54 45 13 20 ef e7 a5 1e 3c d7 b0
  d3 3d 87 65 0e 69 c2 76 2e 86 25 d3 30 ba f2 63
○ mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ ▯
```

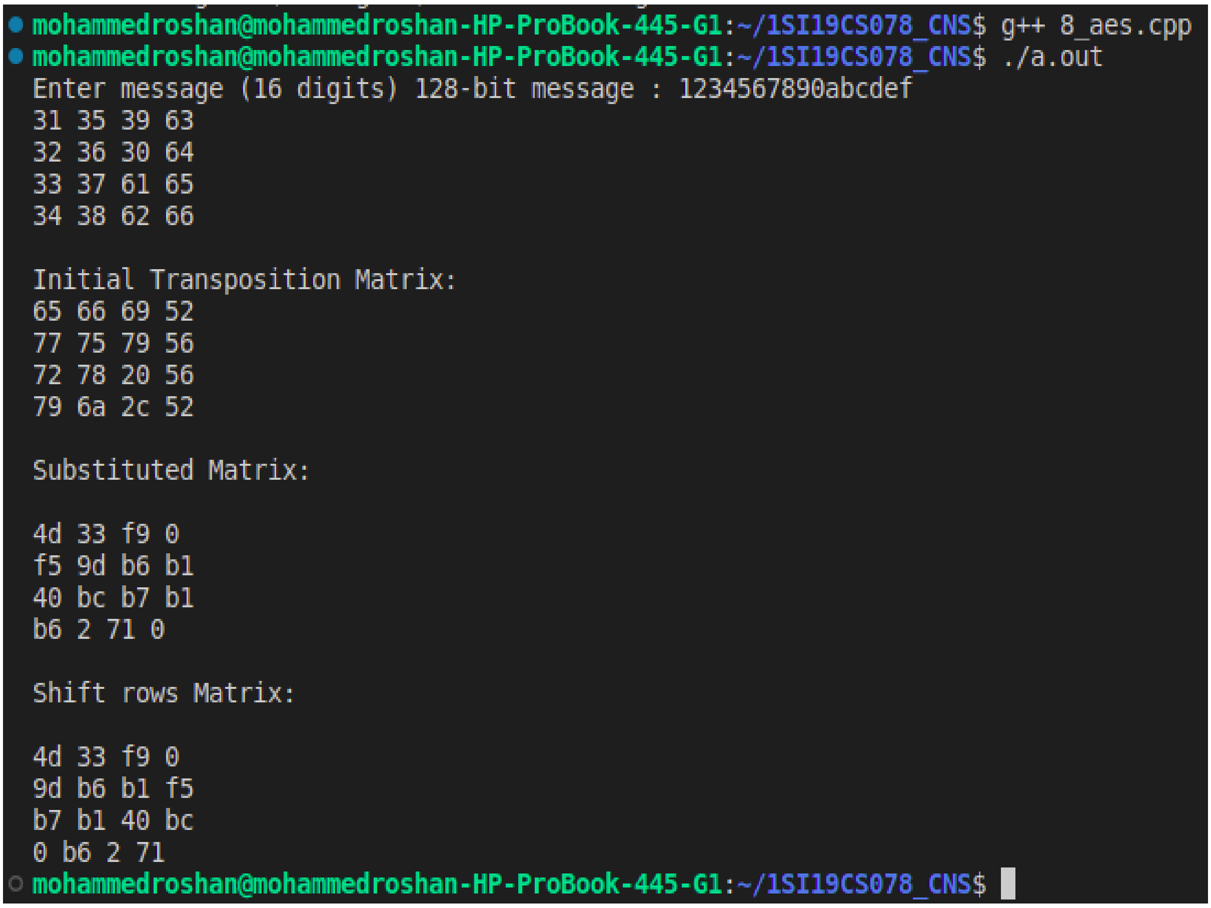
8. Consider a message of 16 bytes (128 bits) and perform XOR operation with an initial round key [W0, W1, W2, W3] of size 128 bits to generate a state array in AES. W.r.t generated state array of size 128 bits, perform the following operations in each round.
i.Byte substitution using S-Box
ii.ShiftRows using left shift

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 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++)
  {
    if(x[i]==y[i]) res+="0";
    else res+="1";
  }
  return res;
}
string Sbox_subst(string input){
        int row=bitset<4>(input.substr(0,4)).to_ulong();
        int col=bitset<4>(input.substr(4,4)).to_ulong();
        return bitset<8>(sbox[row][col]).to_string();
}
int main(){
        string msg;
        cout << "Enter message (16 digits) 128-bit message : ";
        cin >> msg;
        char dec_to_hex[16]={'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
        string mat[4][4];
        for(int i=0;i<16;i++){
            int ascii=msg[i];
            string hex="";
            while(ascii!=0){
                hex=dec_to_hex[(ascii%16)]+hex;
                ascii=ascii/16;
            }
            mat[i%4][i/4]=hex;
        }
        for(int i=0;i<4;i++){
            for(int j=0;j<4;j++){
                cout<<mat[i][j]<<" ";
            }
        cout<<endl;
        }
        string init[4][4];
        cout << "\nInitial Transposition Matrix:\n";
        for(int i=0;i<4;i++){
         for(int j=0;j<4;j++){
            int val=stoi(mat[i][j],0,16);
            string temp1 = bitset<8>(val).to_string();
            string temp2=bitset<8>(key[i][j]).to_string();
            init[i][j]=XOR(temp1,temp2);

            cout << hex<< bitset<8>(init[i][j]).to_ulong() <<" ";
        }
        cout<<endl;
  }
         cout << "\nSubstituted Matrix:\n";
        cout<<endl;
        string subst[4][4];
         for(int i=0;i<4;i++){
            for(int j=0;j<4;j++){
               subst[i][j]=Sbox_subst(init[i][j]);
               cout<<hex<< bitset<8>(subst[i][j]).to_ulong() <<" ";
            }
        cout << endl;
        }
```

```
            cout << "\nShift rows Matrix:\n";
            cout<<endl;
            for(int i=0;i<4;i++){
                        for(int j=0;j<4;j++){
                            cout<<hex<< bitset<8>(subst[i][(j+i)%4]).to_ulong() <<" ";
                        }
            cout << endl;
            }
   return 0;
}
```

Output :



```
● mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ g++ 8_aes.cpp
● mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ ./a.out
  Enter message (16 digits) 128-bit message : 1234567890abcdef
  31 35 39 63
  32 36 30 64
  33 37 61 65
  34 38 62 66

  Initial Transposition Matrix:
  65 66 69 52
  77 75 79 56
  72 78 20 56
  79 6a 2c 52

  Substituted Matrix:

  4d 33 f9 0
  f5 9d b6 b1
  40 bc b7 b1
  b6 2 71 0

  Shift rows Matrix:

  4d 33 f9 0
  9d b6 b1 f5
  b7 b1 40 bc
  0 b6 2 71
○ mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$
```

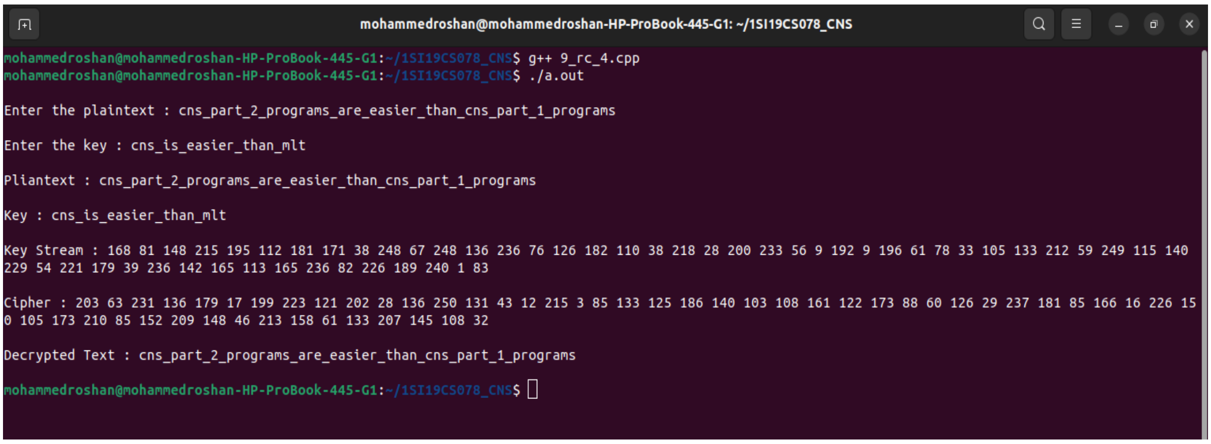
9.Implement the following with respect to RC4:

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

Code :

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

int main()
{
        string plaintext,key;
        cout<<"\nEnter the plaintext : ";
        cin>>plaintext;
        cout<<"\nEnter the key : ";
        cin>>key;
        cout<<endl;
        cout<<"Pliantext : "<<plaintext<<endl;
        cout<<endl<<"Key : "<<key<<endl;
        int S[256],T[256],keyStream[256],cipher[256];
        for(int i=0;i<256;i++){
                S[i]=i;
                T[i]=key[(i%key.length())];

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

        cout<<endl<<"Key Stream : ";

        j=0;
        for(int i=0;i<plaintext.length();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<<endl;
        cout<<endl<<"Cipher : ";
        for(int i=0;i<plaintext.length();i++){
                cipher[i]=plaintext[i]^keyStream[i];
                cout << cipher[i] <<" ";
        }
        cout<<endl;
        cout<<endl<<"Decrypted Text : ";
        for(int i=0;i<plaintext.length();i++){
                plaintext[i]=cipher[i]^keyStream[i];
                cout << plaintext[i];
        }
        cout<<endl<<endl;
        return 0;
}
```

Output :



mohammedroshan@mohammedroshan-HP-ProBook-445-G1: ~/1SI19CS078_CNS

```
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ g++ 9_rc_4.cpp
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ ./a.out

Enter the plaintext : cns_part_2_programs_are_easier_than_cns_part_1_programs

Enter the key : cns_is_easier_than_mlt

Pliantext : cns_part_2_programs_are_easier_than_cns_part_1_programs

Key : cns_is_easier_than_mlt

Key Stream : 168 81 148 215 195 112 181 171 38 248 67 248 136 236 76 126 182 110 38 218 28 200 233 56 9 192 9 196 61 78 33 105 133 212 59 249 115 140
229 54 221 179 39 236 142 165 113 165 236 82 226 189 240 1 83

Cipher : 203 63 231 136 179 17 199 223 121 202 28 136 250 131 43 12 215 3 85 133 125 186 140 103 108 161 122 173 88 60 126 29 237 181 85 166 16 226 15
0 105 173 210 85 152 209 148 46 213 158 61 133 207 145 108 32

Decrypted Text : cns_part_2_programs_are_easier_than_cns_part_1_programs

mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ 
```

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.

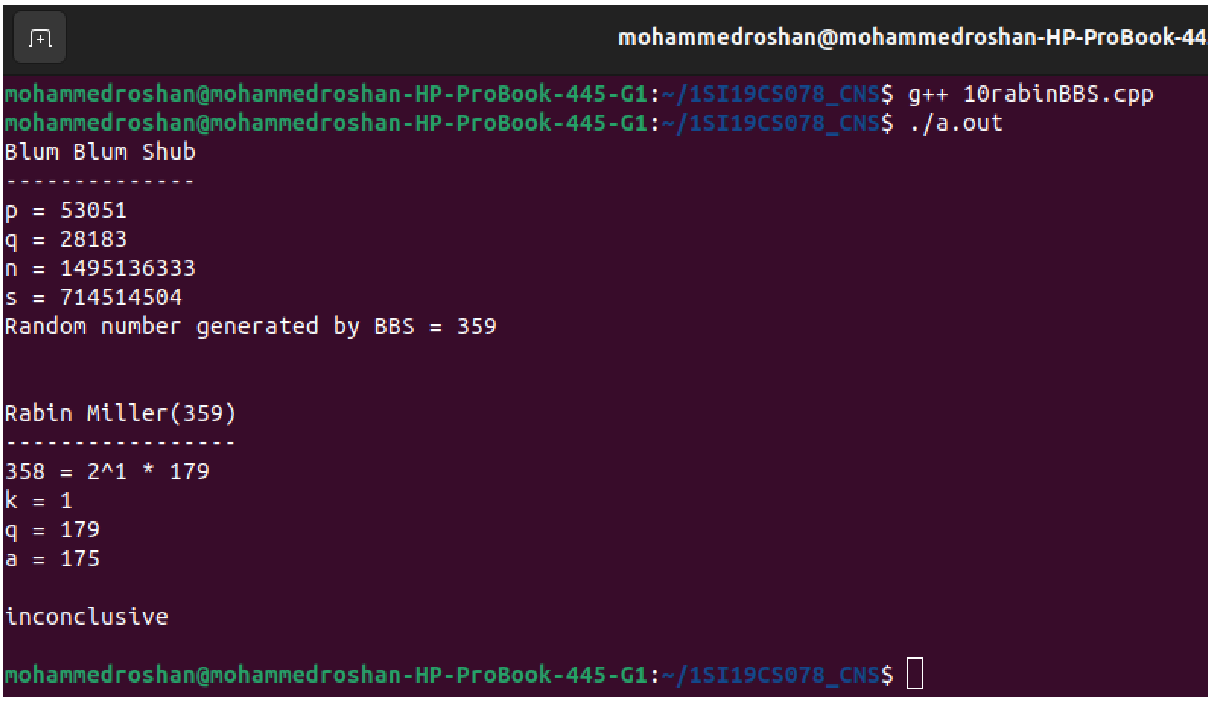
Code :

```
#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<<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<<endl;

        cout<<rabinMiller(randNum) << endl<<endl;

        return 0;
}
```

Output :



```
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ g++ 10rabinBBS.cpp
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ ./a.out
Blum Blum Shub
--------------
p = 53051
q = 28183
n = 1495136333
s = 714514504
Random number generated by BBS = 359


Rabin Miller(359)
-----------------
358 = 2^1 * 179
k = 1
q = 179
a = 175

inconclusive

mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ 
```

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.

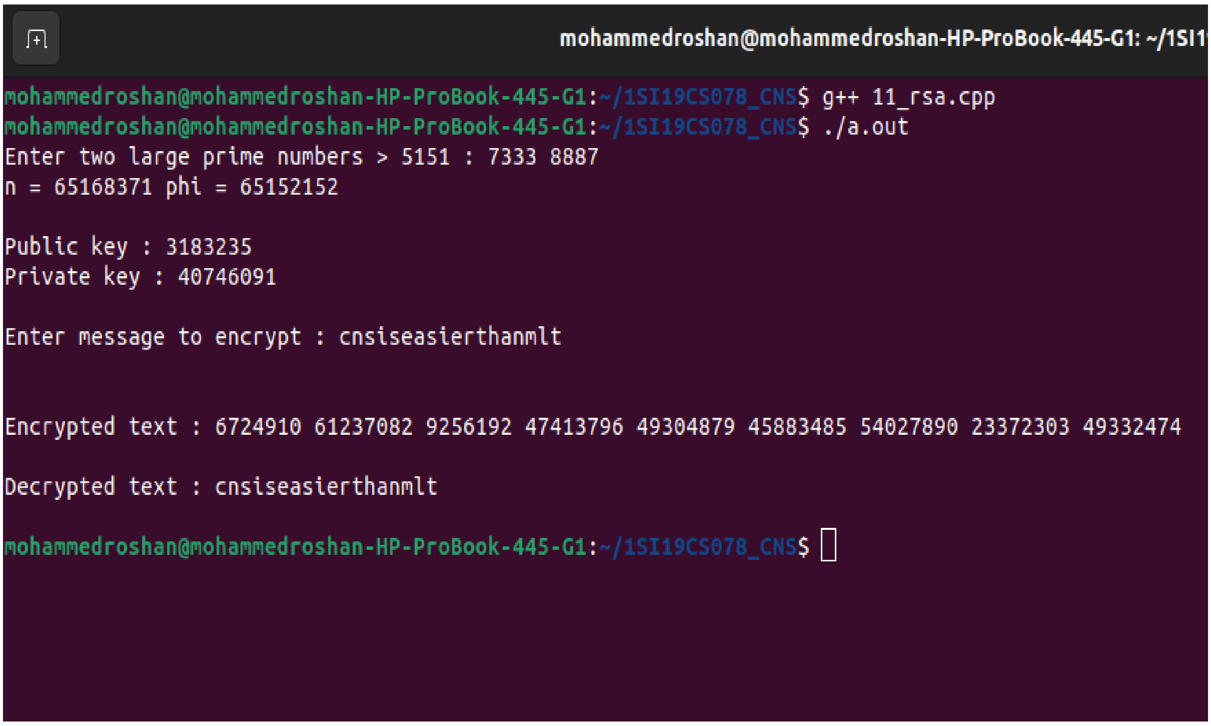
iv. Display the result after each step.

Code :

```
#include <bits/stdc++.h>
using namespace std;
long long randInRange(long long low, long long high)
{
   return rand()%(high-(low-1)) + (low+1) ;
}

long long gcd(long long a, long long b)
{
   if(b==0) return a;
   return gcd(b, a%b);
}
long long powermod(long long a, long long b, long long n)
{
   long long res = 1;
   for(long long i=0; i<b; i++)
   {
      res = (res*a) % n;
   }
   return res;
}
long long decrypt(long long C,long long d,long long n)
{
   return powermod(C,d,n);
}
long long encrypt(long long M, long long e,long long n)
{
   return powermod(M,e,n);
}

int main()
{

          long long p,q;
```

```
        cout<<"Enter two large prime numbers > 5151 : ";
        cin>>p>>q;
        long long n=p*q,e;
        long long phi=(p-1)*(q-1);
        cout<<"n = "<<n<<" phi = "<<phi<<endl<<endl;
        e=randInRange(1,phi);
        while(gcd(e,phi)!=1){
                e=randInRange(1,phi);
        }
        long long d=1;
        while(((long long)d*e)%phi!=1){
                d+=1;
        }
        cout<<"Public key : "<<e<<endl;
        cout<<"Private key : "<<d<<endl;
        map<char,long long> m1;
        map<long long,char> m2;
        for(char ch='a';ch<='z';ch++){
                m1[ch]=ch-'a';
                m2[ch-'a']=ch;
                m1[ch-32]=ch-'a'+26;
                m2[ch-'a'+26]=ch-32;
        }
        string msg;
        cout << "\nEnter message to encrypt : ";
        cin >> msg;
        cout<<endl<<endl;
        if(msg.length()% 2 != 0) msg+="x";
        vector<long long> enc_text;
        cout<<"Encrypted text : ";
        for(long long i=0;i<msg.length();i+=2){
                long long M=m1[msg[i]]*100+m1[msg[i+1]];
                long long C=encrypt(M,e,n);
                enc_text.push_back(C);
                cout<<C<<" ";
        }
        cout<<endl<<endl;
        cout<<"Decrypted text : ";
        for(long long i=0;i<enc_text.size();i++){
                long long M=decrypt(enc_text[i],d,n);
                cout<<m2[M/100]<<m2[M%100];
        }
        cout<<endl<<endl;
        return 0;
}
```

Output :



```
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ g++ 11_rsa.cpp
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ ./a.out
Enter two large prime numbers > 5151 : 7333 8887
n = 65168371 phi = 65152152

Public key : 3183235
Private key : 40746091

Enter message to encrypt : cnsiseasierthanmlt


Encrypted text : 6724910 61237082 9256192 47413796 49304879 45883485 54027890 23372303 49332474

Decrypted text : cnsiseasierthanmlt

mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ 
```

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

Code :

Server's Program :

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

using namespace std;

int PUs[2], PRs[2],temp[2];

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

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

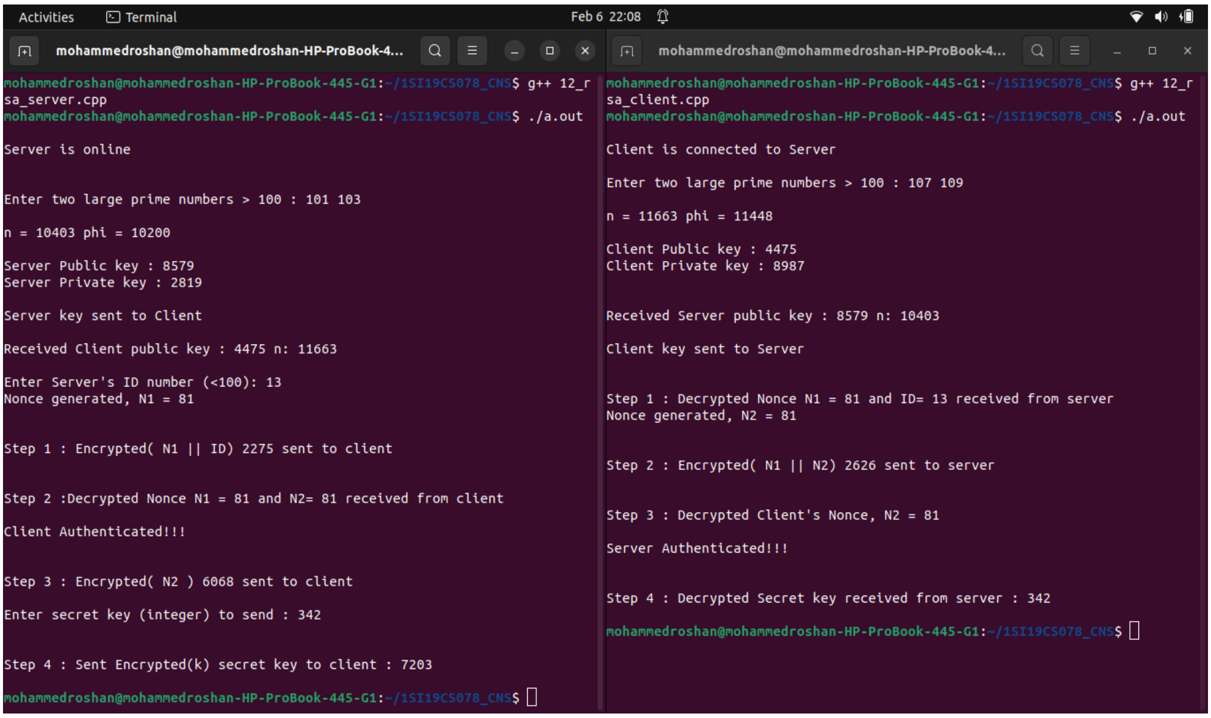
```
}

int main()
{
        int sersock, sock;
        sersock=socket(AF_INET,SOCK_STREAM,0);
        struct sockaddr_in addr = { AF_INET, htons(1234), inet_addr("127.0.0.1") };
        // Forcefully connecting to same port everytime
        int reuse = 1;

        cout<<"\nServer is online\n\n";
        setsockopt(sersock, SOL_SOCKET, SO_REUSEADDR, (char *)&reuse, sizeof(reuse));
        /* attaching socket to port */
        bind(sersock, (struct sockaddr *) &addr, sizeof(addr));
        listen(sersock, 5); // listen(int sockfd, int backlog)
        sock = accept(sersock, NULL, NULL);

        int p,q;
        cout<<"\nEnter two large prime numbers > 100 : ";
        cin>>p>>q;
        int n=p*q,e;
        int phi=(p-1)*(q-1);
        cout<<"\nn = "<<n<<" phi = "<<phi<<endl<<endl;
        e=randInRange(1,phi);
        while(gcd(e,phi)!=1){
                e=randInRange(1,phi);
        }
        int d=1;
        while((d*e)%phi!=1){
                d+=1;
        }
        cout<<"Server Public key : "<<e<<endl;
        cout<<"Server Private key : "<<d<<endl<<endl;

        PUs[0]=e;
        PUs[1]=n;
        PRs[0]=d;
        PRs[1]=n;

        send(sock,&PUs,sizeof(PUs),0);
        cout<<"Server key sent to Client"<<endl;

        recv(sock,&temp,sizeof(temp),0);
        cout<<"\nReceived Client public key : "<<temp[0]<<" n: "<<temp[1]<<endl;

        int ID;
        cout<<"\nEnter Server's ID number (<100): ";
        cin>>ID;

        srand(time(NULL));
        int N1 = rand()%100; // nonce
        cout << "Nonce generated, N1 = " << N1 << endl;


        int msg = N1*100 + ID; // append ID to nonce
        int cipher = encrypt(msg,temp);

        send(sock,&cipher,sizeof(cipher),0);

        cout<<"\n\nStep 1 : Encrypted( N1 || ID) "<<cipher<<" sent to client\n";

        recv(sock,&cipher,sizeof(cipher),0);
        msg=decrypt(cipher,PRs);
        int N1c=msg/100;
        int N2=msg%100;

    cout<<"\n\nStep 2 :Decrypted Nonce N1 = "<<N1<<" and N2= "<<N2<<" received from client\n";

        if(N1==N1c) cout<<"\nClient Authenticated!!!"<<endl;
        else { cout<<"\nNonce didn't match, Client Not Authenticated!!!"<<endl; exit(0); }

        cipher = encrypt(N2,temp);

        send(sock,&cipher,sizeof(cipher),0);

        cout<<"\n\nStep 3 : Encrypted( N2 ) "<<cipher<<" sent to client\n";

    int k;
    cout << "\nEnter secret key (integer) to send : ";
    cin >> k;
    cipher = encrypt(encrypt(k,PRs),temp);
    send(sock, &cipher, sizeof(cipher), 0);
    cout << "\n\nStep 4 : Sent Encrypted(k) secret key to client : " << cipher << endl << endl;

return 0;

}
```

Client's Program

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

int PUs[2], PRs[2],temp[2];

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

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

int main()
{
         int sock;
         sock = socket(AF_INET, SOCK_STREAM, 0);
         struct sockaddr_in addr = { AF_INET, htons(1234), inet_addr("127.0.0.1") };
         /* keep trying to esatablish connection with server */
         while(connect(sock, (struct sockaddr *) &addr, sizeof(addr)) < 0) ;
         printf("\nClient is connected to Server\n\n");
```

```
        int p,q;
        cout<<"Enter two large prime numbers > 100 : ";
        cin>>p>>q;
        int n=p*q,e;
        int phi=(p-1)*(q-1);
        cout<<"\nn = "<<n<<" phi = "<<phi<<endl<<endl;
        e=randInRange(1,phi);
        while(gcd(e,phi)!=1){
                e=randInRange(1,phi);
        }
        int d=1;
        while((d*e)%phi!=1){
                d+=1;
        }
        cout<<"Client Public key : "<<e<<endl;
        cout<<"Client Private key : "<<d<<endl<<endl;

        PUs[0]=e;
        PUs[1]=n;
        PRs[0]=d;
        PRs[1]=n;

        recv(sock,&temp,sizeof(temp),0);

        cout<<"\nReceived Server public key : "<<temp[0]<<" n: "<<temp[1]<<endl;

        send(sock,&PUs,sizeof(PUs),0);
        cout<<"\nClient key sent to Server"<<endl;

        int cipher;
        recv(sock,&cipher,sizeof(cipher),0);
        int msg=decrypt(cipher,PRs);
        int N1=msg/100;
        int ID=msg%100;

   cout<<"\n\nStep 1 : Decrypted Nonce N1 = "<<N1<<" and ID= "<<ID<<" received from server\n";
        srand(time(NULL));
        int N2 = rand()%100; // nonce
        cout << "Nonce generated, N2 = " << N2 << endl;

        msg = N1*100 + N2;
        cipher = encrypt(msg,temp);

        send(sock,&cipher,sizeof(cipher),0);
        cout<<"\n\nStep 2 : Encrypted( N1 || N2) "<<cipher<<" sent to server\n";
        recv(sock,&cipher,sizeof(cipher),0);
        int N2c=decrypt(cipher,PRs);
        cout << "\n\nStep 3 : Decrypted Client's Nonce, N2 = " << N2c << endl;
        if(N2==N2c) cout<<"\nServer Authenticated!!!"<<endl;
        else { cout<<"\nNonce didn't match, Server Not Authenticated!!!"<<endl; exit(0); }

        recv(sock,&cipher,sizeof(cipher),0);
        int k=decrypt(decrypt(cipher,PRs),temp);

        cout<<"\n\nStep 4 : Decrypted Secret key received from server : "<<k<<endl<<endl;

        return 0;
}
```

<u>Output</u> :



Activities
Terminal
Feb 6 22:08
mohammedroshan@mohammedroshan-HP-ProBook-4...
mohammedroshan@mohammedroshan-HP-ProBook-4...

```
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ g++ 12_r
sa_server.cpp
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ ./a.out

Server is online


Enter two large prime numbers > 100 : 101 103

n = 10403 phi = 10200

Server Public key : 8579
Server Private key : 2819

Server key sent to Client

Received Client public key : 4475 n: 11663

Enter Server's ID number (<100): 13
Nonce generated, N1 = 81


Step 1 : Encrypted( N1 || ID) 2275 sent to client


Step 2 :Decrypted Nonce N1 = 81 and N2= 81 received from client

Client Authenticated!!!


Step 3 : Encrypted( N2 ) 6068 sent to client

Enter secret key (integer) to send : 342


Step 4 : Sent Encrypted(k) secret key to client : 7203

mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ 
```

```
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ g++ 12_r
sa_client.cpp
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ ./a.out

Client is connected to Server

Enter two large prime numbers > 100 : 107 109

n = 11663 phi = 11448

Client Public key : 4475
Client Private key : 8987


Received Server public key : 8579 n: 10403

Client key sent to Server


Step 1 : Decrypted Nonce N1 = 81 and ID= 13 received from server
Nonce generated, N2 = 81


Step 2 : Encrypted( N1 || N2) 2626 sent to server


Step 3 : Decrypted Client's Nonce, N2 = 81

Server Authenticated!!!


Step 4 : Decrypted Secret key received from server : 342

mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ 
```

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) .

Code :

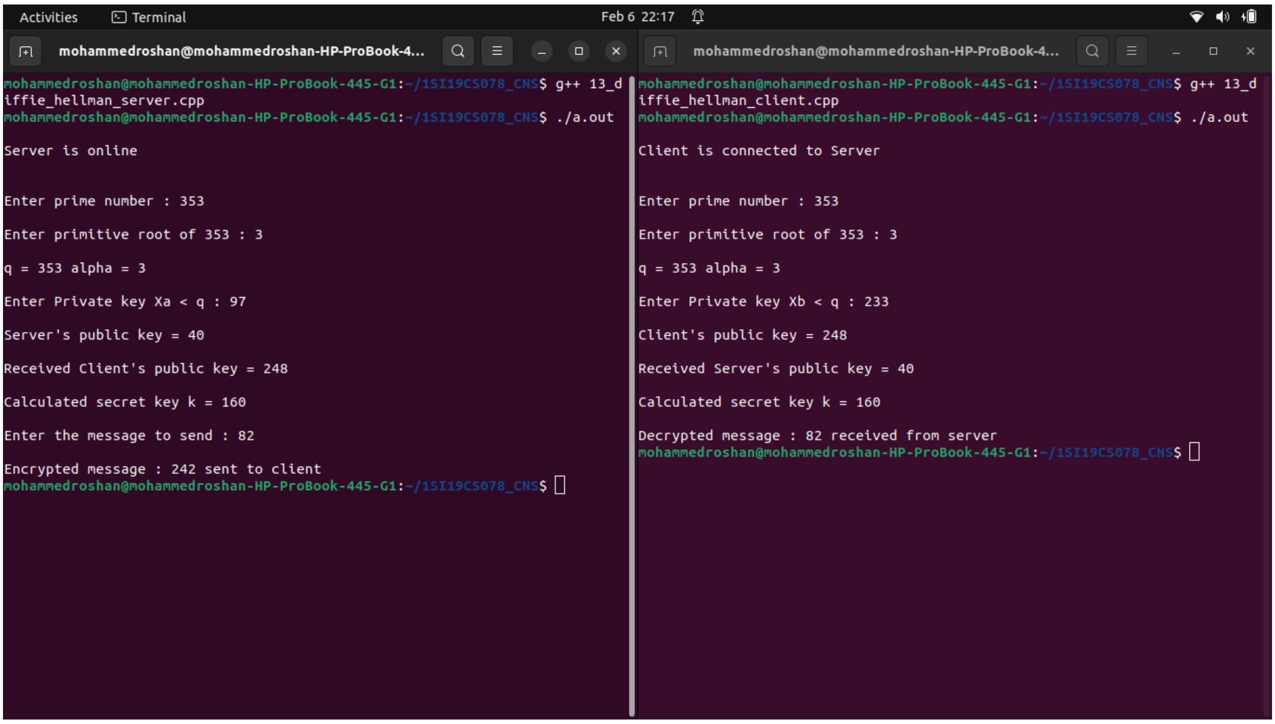
Server's Program

```
#include<bits/stdc++.h>
#include<arpa/inet.h>
using namespace std;
int pow_mod_n(int a,int b,int n){
        int res=1;
        for(int i=0;i<b;i++){
                res=(res*a)%n;
        }
        return res;
}
int main(){
        int sock=socket(AF_INET,SOCK_STREAM,0);
        int reuse=1;
        setsockopt(sock,SOL_SOCKET,SO_REUSEADDR,(char *) &reuse,sizeof(reuse));

        cout<<"\nServer is online\n\n";

        struct sockaddr_in addr={AF_INET,htons(1234),inet_addr("127.0.0.1")};
        bind(sock,(struct sockaddr*)&addr,sizeof(addr));
        listen(sock,5);
        sock=accept(sock,NULL,NULL);

        int q,alpha;
        int Xa,Ya,Yb;

        cout<<"\nEnter prime number : ";
        cin>>q;
        cout<<"\nEnter primitive root of "<<q<<" : ";
        cin>>alpha;
        cout<<"\nq = "<<q<<" alpha = "<<alpha<<endl;

        cout<<"\nEnter Private key Xa < q : ";
        cin>>Xa;
        Ya=pow_mod_n(alpha,Xa,q);
        cout<<"\nServer's public key = "<<Ya<<endl;
        send(sock,&Ya,sizeof(Ya),0);
        recv(sock,&Yb,sizeof(Yb),0);
        cout<<"\nReceived Client's public key = "<<Yb<<endl;
        int k=pow_mod_n(Yb,Xa,q);
        cout<<"\nCalculated secret key k = "<<k<<endl;
        int msg;
        cout<<"\nEnter the message to send : ";
        cin>>msg;
        int encrypt=msg^k;
        send(sock,&encrypt,sizeof(encrypt),0);
        cout<<"\nEncrypted message : "<<encrypt<<" sent to client"<<endl;
        return 0;
}
```

Client’s Program

```
#include<bits/stdc++.h>
#include<arpa/inet.h>
using namespace std;
int pow_mod_n(int a,int b,int n){
        int res=1;
        for(int i=0;i<b;i++){
                res=(res*a)%n;
        }
        return res;
}
int main(){
        int sock=socket(AF_INET,SOCK_STREAM,0);
        struct sockaddr_in addr={AF_INET,htons(1234),inet_addr("127.0.0.1")};
        while(connect(sock,(struct sockaddr*)&addr,sizeof(addr))<0);

        cout<<"\nClient is connected to Server\n\n";

        int q,alpha;
        int Xb,Yb,Ya;

        cout<<"\nEnter prime number : ";
        cin>>q;
        cout<<"\nEnter primitive root of "<<q<<" : ";
        cin>>alpha;
        cout<<"\nq = "<<q<<" alpha = "<<alpha<<endl;

        cout<<"\nEnter Private key Xb < q : ";
        cin>>Xb;
        Yb=pow_mod_n(alpha,Xb,q);
        cout<<"\nClient's public key = "<<Yb<<endl;
        recv(sock,&Ya,sizeof(Ya),0);
        send(sock,&Yb,sizeof(Yb),0);
        cout<<"\nReceived Server's public key = "<<Ya<<endl;
        int k=pow_mod_n(Ya,Xb,q);
        cout<<"\nCalculated secret key k = "<<k<<endl;
        int encrypt;
        recv(sock,&encrypt,sizeof(encrypt),0);
        int msg=encrypt^k;
        cout<<"\nDecrypted message : "<<msg<<" received from server"<<endl;
        return 0;
}
```

Output :



Activities
Terminal
Feb 6 22:17
mohammedroshan@mohammedroshan-HP-ProBook-4...
mohammedroshan@mohammedroshan-HP-ProBook-4...

```
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ g++ 13_d
iffie_hellman_server.cpp
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ ./a.out

Server is online


Enter prime number : 353

Enter primitive root of 353 : 3

q = 353 alpha = 3

Enter Private key Xa < q : 97

Server's public key = 40

Received Client's public key = 248

Calculated secret key k = 160

Enter the message to send : 82

Encrypted message : 242 sent to client
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ 
```

```
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ g++ 13_d
iffie_hellman_client.cpp
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ ./a.out

Client is connected to Server


Enter prime number : 353

Enter primitive root of 353 : 3

q = 353 alpha = 3

Enter Private key Xb < q : 233

Client's public key = 248

Received Server's public key = 40

Calculated secret key k = 160

Decrypted message : 82 received from server
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ 
```

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).

Code :

Server’s Program

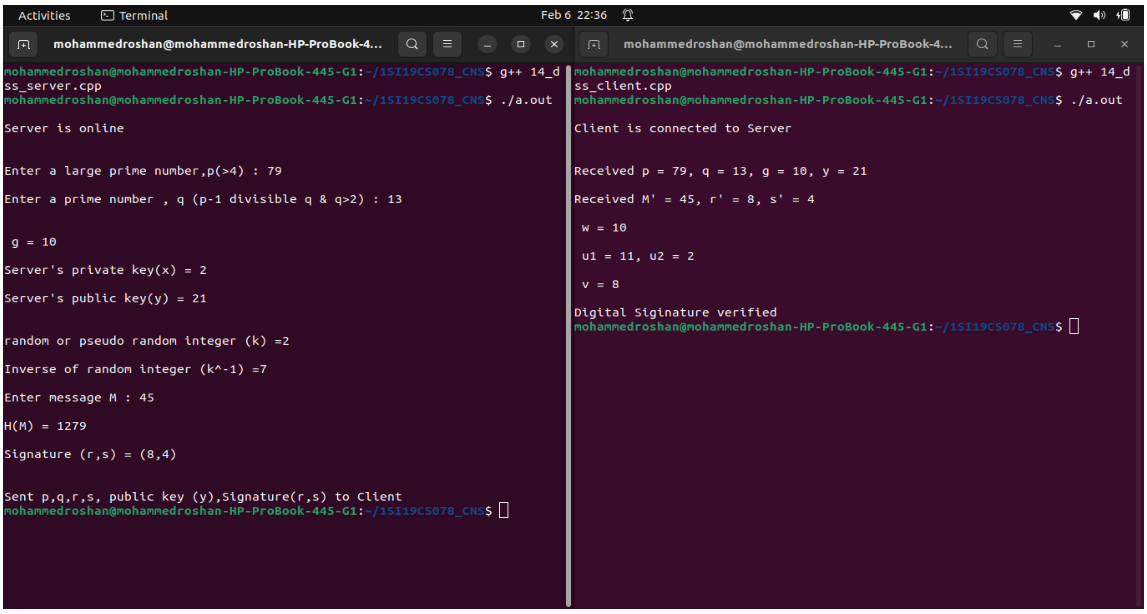
```
#include<bits/stdc++.h>
#include<arpa/inet.h>
using namespace std;
long long randInRange(long long low, long long high){
  return rand()%(high-(low-1)) + (low+1) ;
}
int pow_mod_n(int a,int b,int n){
        int res=1;
        for(int i=0;i<b;i++)
                res=(res*a)%n;
        return res;
}
int Hash(int M){
        return M^1234;
}
int main(){
        int sock=socket(AF_INET,SOCK_STREAM,0);
        int reuse=1;
        setsockopt(sock,SOL_SOCKET,SO_REUSEADDR,(char *) &reuse,sizeof(reuse));
        cout<<"\nServer is online\n\n";
        struct sockaddr_in addr={AF_INET,htons(1234),inet_addr("127.0.0.1")};
        bind(sock,(struct sockaddr*)&addr,sizeof(addr));
        listen(sock,5);
        sock=accept(sock,NULL,NULL);

        int p,q;
        cout<<"\nEnter a large prime number,p(>4) : ";
        cin>>p;
        cout<<"\nEnter a prime number , q (p-1 divisible q & q>2) : ";
        cin>>q;
        if((p-1)%q!=0 || q<=2) {
                cout<<"\nInvalid Input\n";
                exit(0);
        }
        srand(time(NULL));

        int h=randInRange(1,p-1);
        int g=pow_mod_n(h,(p-1)/q,p);
```

```
    while(g<=1){
        h=randInRange(1,p-1);
        g=pow_mod_n(h,(p-1)/q,p);
    }
    cout<<"\n\n g = "<<g<<endl;
    int x=randInRange(0,q);
    cout<<"\nServer's private key(x) = "<<x<<endl;
    int y=pow_mod_n(g,x,p);
    cout<<"\nServer's public key(y) = "<<y<<endl;
    int k=randInRange(0,q);

    cout<<"\n\nrandom or pseudo random integer (k) ="<<k<<endl;
    int inv_k=1;
    while((inv_k*k)%q!=1){
        inv_k+=1;
    }
    cout<<"\nInverse of random integer (k^-1) ="<<inv_k<<endl;
    int M;
    cout<<"\nEnter message M : ";
    cin>>M;
    int H=Hash(M);
    cout<<"\nH(M) = "<<H<<endl;

    int r=pow_mod_n(g,k,p)%q;
    int s=(inv_k*(H+x*r))%q;
    cout<<"\nSignature (r,s) = ("<<r<<","<<s<<")"<<endl;

    int arr[7]={p,q,g,y,M,r,s};
    send(sock,&arr,sizeof(arr),0);
    cout<<"\n\nSent p,q,r,s, public key (y),Signature(r,s) to Client\n";
    return 0;
}
```

Client's Program

```
#include<bits/stdc++.h>
#include<arpa/inet.h>
using namespace std;
int pow_mod_n(int a,int b,int n){
    int res=1;
    for(int i=0;i<b;i++){
        res=(res*a)%n;
    }
    return res;
}
int Hash(int M){
    return M^1234;
}
int main(){
    int sock=socket(AF_INET,SOCK_STREAM,0);
    struct sockaddr_in addr={AF_INET,htons(1234),inet_addr("127.0.0.1")};
    while(connect(sock,(struct sockaddr*)&addr,sizeof(addr))<0);

    cout<<"\nClient is connected to Server\n\n";
    int arr[7];
    recv(sock,&arr,sizeof(arr),0);
    int p=arr[0],q=arr[1],g=arr[2],y=arr[3];
    int M=arr[4],r=arr[5],s=arr[6];
    cout<<"\nReceived p = "<<p<<", q = "<<q<<", g = "<<g<<", y = "<<y<<endl;
    cout<<"\nReceived M' = "<<M<<", r' = "<<r<<", s' = "<<s<<endl;
    int w=1;
    while((s*w)%q!=1) w+=1;
    cout<<"\n w = "<<w<<endl;
    int u1=(Hash(M)*w)%q;
    int u2=(r*w)%q;
    cout<<"\n u1 = "<<u1<<", u2 = "<<u2<<endl;
    int v=((pow_mod_n(g,u1,p)*pow_mod_n(y,u2,p))%p)%q;
    cout<<"\n v = "<<v<<endl;

    if(v==r)
        cout<<"\nDigital Siginature verified\n";
    else
        cout<<"\nDigital Siginature not verified\n";

    return 0;
}
```

Output :


Activities
Terminal
Feb 6 22:36
mohammedroshan@mohammedroshan-HP-ProBook-4...
mohammedroshan@mohammedroshan-HP-ProBook-4...

```
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ g++ 14_d
ss_server.cpp
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ ./a.out

Server is online


Enter a large prime number,p(>4) : 79

Enter a prime number , q (p-1 divisible q & q>2) : 13


 g = 10

Server's private key(x) = 2

Server's public key(y) = 21


random or pseudo random integer (k) =2

Inverse of random integer (k^-1) =7

Enter message M : 45

H(M) = 1279

Signature (r,s) = (8,4)


Sent p,q,r,s, public key (y),Signature(r,s) to Client
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ 
```

```
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ g++ 14_d
ss_client.cpp
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ ./a.out

Client is connected to Server


Received p = 79, q = 13, g = 10, y = 21

Received M' = 45, r' = 8, s' = 4

 w = 10

 u1 = 11, u2 = 2

 v = 8

Digital Siginature verified
mohammedroshan@mohammedroshan-HP-ProBook-445-G1:~/1SI19CS078_CNS$ 
```