```
//M.V.HARSHITHA
//rsa implementation
//20134117
#include<bits/stdc++.h>

using namespace std;

long int p,q,d,phi,n,e;

bool prime(long int p)

{

        long int i=2;

        for(i=2;i<p;i++)

        if(p%i==0)

        return false;

        return true;

}

long int pwr(long int a,long int b)

{

        if(b==0)

        return 1;

        if(b==1)

        return a;

        long int temp=pwr(a,b/2);

        if(b%2==0)

        return (temp*temp)%n;

        else

        return ((temp*temp)%n*a)%n;

}

int main()

{
```

```
cout<<"enter value of p and q"<<endl;

cin>>p>>q;

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

n=p*q;

for(int i=2;i<phi;i++)

{

        if(i!=p&&i!=q&&prime(i))

        {

                for(int j=2;j<phi;j++)

                if((i*j)% phi==1)

                {

                        e=i;

                        d=j;

                        i=phi+1;

                        j=phi+1;

                }

        }

}

cout<<pwr('x',e)%255<<'\t';

cout<<pwr(pwr('x',e),d)<<endl;

cout<<"enter message..."<<endl;

string a;

cin>>a;

string cipher="",msg="";

int len=a.size();
```

```
        for(int i=0;i<len;i++)

        {

                cipher=cipher+(char)((pwr(a[i]-'a',e) % n));

        }

        cout<<"decrypted message.."<<endl;

        cout<<cipher<<endl;

        for(int i=0;i<len;i++)

        {

                msg=msg+(char)((pwr(cipher[i]-'a',d) % n));

                //cout<<msg<<'\t';

        }

        cout<<"enrypted value.."<<endl;

        cout<<msg<<endl;

        cout<<e<<'\t'<<d<<endl;

}
```

```
//M.V.HARSHITHA
//DES

import java.security.spec.KeySpec;
import java.io.*;
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.DESKeySpec;
import sun.misc.BASE64Decoder;
import sun.misc.BASE64Encoder;
 import java.util.*;
public class DES {

    public static final String DES_ENCRYPTION_SCHEME = "DES";
    private KeySpec myKeySpec;
    private SecretKeyFactory mySecretKeyFactory;
    private Cipher cipher;
    byte[] keyAsBytes;
    private String myEncryptionKey;
private String myEncryptionScheme;

SecretKey key;


public DES() throws Exception

{

    myEncryptionKey = "ThisIsSecretEncryptionKey";

    myEncryptionScheme = DES_ENCRYPTION_SCHEME;

    keyAsBytes = myEncryptionKey.getBytes();

    myKeySpec = new DESKeySpec(keyAsBytes);

    mySecretKeyFactory = SecretKeyFactory.getInstance(myEncryptionScheme);

    cipher = Cipher.getInstance(myEncryptionScheme);

    key = mySecretKeyFactory.generateSecret(myKeySpec);

}


/*Method To Encrypt The String

    */

public String encrypt(String unencryptedString) {

    String encryptedString = null;

    try {

        cipher.init(Cipher.ENCRYPT_MODE, key);

        byte[] plainText = unencryptedString.getBytes();//UNICODE_FORMAT);

        byte[] encryptedText = cipher.doFinal(plainText);

        BASE64Encoder base64encoder = new BASE64Encoder();

        encryptedString = base64encoder.encode(encryptedText);

    } catch (Exception e) {

        e.printStackTrace();
```

```
       }
       return encryptedString;
   }
   /**
*       * Method To Decrypt An Ecrypted String
*            */
   public String decrypt(String encryptedString) {
       String decryptedText=null;
       try {
           cipher.init(Cipher.DECRYPT_MODE, key);
           BASE64Decoder base64decoder = new BASE64Decoder();
           byte[] encryptedText = base64decoder.decodeBuffer(encryptedString);
           byte[] plainText = cipher.doFinal(encryptedText);
           decryptedText= bytes2String(plainText);
       } catch (Exception e) {
           e.printStackTrace();
       }
       return decryptedText;
   }
   /**
*       * Returns String From An Array Of Bytes
*            */
   private static String bytes2String(byte[] bytes) {
       StringBuffer stringBuffer = new StringBuffer();
       for (int i = 0; i< bytes.length; i++) {
           stringBuffer.append((char) bytes[i]);
```

```
        }
        return stringBuffer.toString();
    }

    /**
 *      * Testing the DES Encryption And Decryption Technique
 *           */
    public static void main(String args []) throws Exception
    {
        DES myEncryptor= new DES();
    while (true) {
            String stringToEncrypt="";
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
         stringToEncrypt=br.readLine();
        String encrypted=myEncryptor.encrypt(stringToEncrypt);
            String decrypted=myEncryptor.decrypt(encrypted);

                System.out.println("String To Encrypt: "+stringToEncrypt);
            System.out.println("Encrypted Value :" + encrypted);
            System.out.println("Decrypted Value :"+decrypted);
     }
    }

}
```

```
//hill cipher encryption
//M.V.HARSHITHA
//20134117
using namespace std;

#include <bits/stdc++.h>

int a[10][10],b[9][9];

int main()

{

    cout<<"enter the message\n";

    string s;

    cin>>s;

    cout<<"enter the key matrix of size 3\n";

    int i,j,k,l,m,n;

    string p;

    cin>>p;

    k=0;

    for(i=0;i<3;i++)

    for(j=0;j<3;j++)

    a[i][j]=p[k++]-97;

    n=s.length();

    if(n%3)

    {

        s+='x';

        n++;

    }

    if(n%3)
```

```
{
    s+='x';
    n++;
}
cout<<s<<"\n";
n=s.length();
cout<<"the encrypted text is \n";
for(i=0;i<n;i++)
{
    b[0][0]=s[i]-97;
    b[1][0]=s[++i]-97;
    b[2][0]=s[++i]-97;
    for(l=0;l<3;l++)
    {

        for(j=0;j<1;j++)
        {
            int sum=0;
            for(k=0;k<3;k++)
            sum=sum+a[l][k]*b[k][j];
            cout<<(char)(sum%26+97);

        }



    }
    }

    cout<<"\n";

    return 0;



}

//hill decryption
//20134117
//M.V.HARSHITHA
using namespace std;

#include <bits/stdc++.h>

int a[3][3],b[9][9];

int x;

void inverse(int b[][3])

{

int det=0,i,j;

    int a[3][3];

    for(i=0;i<3;i++)

    for(j=0;j<3;j++)

    a[i][j]=b[i][j];

    for(i=0;i<3;i++)

          det = det + (a[0][i]*(a[1][(i+1)%3]*a[2][(i+2)%3] - a[1][(i+2)%3]*a[2][(i+1)%3]));

          for(i=0;i<3;i++)

          for(j=0;j<3;j++)

          {

              b[j][i]=((a[(i+1)%3][(j+1)%3] * a[(i+2)%3][(j+2)%3]) - (a[(i+1)%3][(j+2)%3]*a[(i+2)%3]
[(j+1)%3]));

          }
```

```
           det=(det+26)%26;
            for(i=0;i<3;i++)
    for(j=0;j<3;j++)
    b[i][j]=(b[i][j]%26+26)%26;
           for(x=1;x<=100;x++)
           {
               if((x*det)%26==1)
               break;
           }

          for(i=0;i<3;i++)
    for(j=0;j<3;j++)
    b[i][j]=(b[i][j]*x+26)%26;
}
int main()
{
    cout<<"enter the message\n";
    string s;
    cin>>s;
    cout<<"enter the key matrix of size 3\n";
    int i,j,k,l,m,n;
    string p;
    cin>>p;
    k=0;
    for(i=0;i<3;i++)
    for(j=0;j<3;j++)
```

```
a[i][j]=p[k++]-97;

n=s.length();

if(n%3)

{

   s+='x';

   n++;

}

if(n%3)

{

   s+='x';

   n++;

}

n=s.length();

inverse(a);

if(x==101)

{

   cout<<"key doesnot have a inverse\n";

   return 0;

}

cout<<"the decrypted text is \n";

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

{

   b[0][0]=s[i]-97;

   b[1][0]=s[++i]-97;

   b[2][0]=s[++i]-97;

   for(l=0;l<3;l++)
```

```
        {


            for(j=0;j<1;j++)

            {

                int sum=0;

                for(k=0;k<3;k++)

                sum=sum+a[l][k]*b[k][j];

                cout<<(char)(sum%26+97);


            }




        }

    }

    cout<<"\n";

    return 0;


}





//diffie-hillman key exchange
//M.V.HARSHITHA
//20134117
#include<bits/stdc++.h>

using namespace std;

long long int power(int a,int b,int mod)

{
 long long int t;
 if(b==1)
  return a;
 t=power(a,b/2,mod);
 if(b%2==0)
  return (t*t)%mod;
 else
  return (((t*t)%mod)*a)%mod;
}
int main()
{
   int p,q,a,b,k1,k2,x,y;
   cout<<"enter the prime number and its generator\n";
   cin>>p>>q;
   cout<<"enter the private exponent of sender a:"<<"\n";
   cin>>a;
   x=power(q,a,p);
    cout<<"enter the private exponent of sender b:"<<"\n";
   cin>>b;
   y=power(q,b,p);
   k1=power(y,a,p);
   k2=power(x,b,p);
   if(k1==k2)
   {cout<<"the keys exchanged are equal"<<"\n";
   cout<<"the key:"<<k1;
   }
   else
   {
   cout<<"keys exchanged are different:"<<k1<<" "<<k2;
   }
   }
```

```
//monoalphabetic cipher encryption n decryption
//M.V.HARSHITHA
//20134117
#include<bits/stdc++.h>
using namespace std;
int main()
 {
  int k,n,i=0,x;
  char p;
  char a[50],c[50],d[50];
  cout<<"enter the shift key";
  cin>>k;
  cout<<"enter the string to be encrypted:";
  cin>>a;
  while(a[i]!='\0')
  {
    p=a[i]+k;
    if(p>122)
    p='a'+k-1;
   // cout<<p;
    c[i]=p;
    i++;
  }
  c[i]='\0';
  cout<<"\nthe encrypted text"<<c;
  cout<<"\ndecryption";
  i=0;
  while(c[i]!='\0')
  {
    p=c[i]-k;
    if(p<'a')
   p= 'z'-k+1;
//  cout<<p;
    d[i]=p;
    i++;
}
 d[i]='\0';
 cout<<"\nthe decrypted text:"<<d;
}
```

```
//polyaplhabetic cipher

#include<stdio.h>

#include<string.h>

main()

{

int i,j;

char p[200],e[200],d[200],k[5];


printf("enter the key:");

scanf("%s",&k);


printf("enter the plaintext:");

scanf("%s",&p);


int kl=strlen(k);


int pl=strlen(p);

j=0;

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

{

e[i]=((p[i]-97+k[j]-97)%26)+97;
j=(j+1)%kl;

}

printf("\nencrypted: %s\n",e);



j=0;

int x;

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

{

x=((e[i]-97-(k[j]-97)));

if(x<0)

x=x+26;

d[i]=(char)(x+97);

j=(j+1)%kl;

}

printf("\ndecrypted: %s\n",d);

}
```

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

using namespace std;
```

```
char encrypt(char );
char decrypt(char );

int main()
{
string plaintext,key,encr_text,decr_text;
string plain="abcdefghijklmnopqrstuvwxyz";
cout<< "enter the substition text  : ";
cin >> key ;
cout << "substitution text : " << key << "\n";
cout << "enter plain text to encrypt : " ;
cin >> plaintext;
int i,k,n;
n=plaintext.length();//n:lenth of text
cout << "encrypted text :  ";
for(i=0;i<n;i++) /* loop for encryption of plain text */
{
        char c=plaintext[i];
        k=c-97;
        //cout << k;
        encr_text[i]=key[k];
        cout << encr_text[i];
}
cout << "\n";
cout << "decrypted text : ";
for(i=0;i<n;i++)/* loop for decryption of cipher text */

{

        char c = encr_text[i];

        int j,l;

        for(j=0;j<26;j++)

        {

                if(key[j]==c)

                {

                k=j;

                break;

                }

        }

        decr_text[i]=plain[k];

        cout << decr_text[i];

}

cout << "\n";

}




//columnar encryption
//20134117
//M.V.HARSHITHA
using namespace std;

#include <bits/stdc++.h>

int vis[1001];

char a[1001][1001];

string s,k;
```

```
int main()
{
        int i,j,l,m,n;
        cout<<"enter the string\n";
        getline(cin,s);
        cout<<"enter the key \n";
        cin>>k;
        m=k.length();
        for(i=0;i<m;i++)
                a[0][i]=k[i];
        int num=0;
        for(j=0;j<26;j++)
        for(i=0;i<m;i++)
        {
                if(a[0][i]==(char)(j+97) && vis[i]==0)
                {
                        a[1][i]=97+num++;
                        vis[i]=1;
                }
        }
        j=0,l=1;
        for(i=0;i<s.length();i++)
        {
                if(s[i]==' ')
                continue;
                if(j==0)l++;
```

```
    a[l][j]=s[i];
    j=(j+1)%m;

}
if(j!=0)
for(i=j;i<m;i++)
a[l][i]='x';
for(i=0;i<=l;i++)
{
    for(j=0;j<m;j++)
    cout<<a[i][j]<<" ";
    cout<<"\n";
}
cout<<"the encrypted text is\n";

for(i=0;i<num;i++)
{
    for(j=0;j<m;j++)
    {
        if(a[1][j]==(char)(i+97))
        {
            for(n=2;n<=l;n++)
            cout<<a[n][j];
            num++;
            break;
        }
```

```
                }

        }

        cout<<"\n";

        return 0;

}


//columnar decryption
//20134117
//M.V.HARSHITHA
using namespace std;

#include <bits/stdc++.h>

int vis[1001];

char a[1001][1001];

string s,k;

int main()

{

        int i,j,l,m,n;

        cout<<"enter the encrypted string\n";

        getline(cin,s);

        cout<<"enter the key \n";

        cin>>k;

        m=k.length();

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

                a[0][i]=k[i];

        int num=0;

        l=s.length()/m;

        for(j=0;j<26;j++)

        for(i=0;i<m;i++)
        {
                if(a[0][i]==(char)(j+97) && vis[i]==0)
                {
                        a[1][i]=97+num++;
                        vis[i]=1;
                }
        }
        int     p=0;
        for(i=0;i<num;i++)
        {
                for(j=0;j<m;j++)
                if(a[1][j]==(char)(i+97))
                {
                        for(n=2;n<=l+1;n++)
                        a[n][j]=s[p++];
                }
        }
        for(i=2;i<=l+1;i++)
        for(j=0;j<m;j++)
        cout<<a[i][j];
        cout<<"\n";
        return 0;
}
```

```
//hash code digital signature
import java.util.*;

import java.math.BigInteger;

import java.security.*;

import javax.crypto.*;

import java.security.MessageDigest;



public class Hashc{
        static KeyPairGenerator gen;

        static KeyPair pair;

        public static String getMD5(String input) {

    try{

        MessageDigest md =MessageDigest.getInstance("MD5");

        byte[] messageDigest=md.digest(input.getBytes());

        BigInteger number=new BigInteger(1, messageDigest);

        String hashtext=number.toString(16);

        while(hashtext.length()<32) {

            hashtext="0"+hashtext;

        }

        return hashtext;

    }

    catch(Exception e){}

    return null;

 }

public static SealedObject encrypt(String message){

        try{

                Cipher c=Cipher.getInstance("RSA");

                c.init(Cipher.ENCRYPT_MODE,pair.getPrivate());

                SealedObject encr=new SealedObject(message,c);

                System.out.println("Encryption as : "+encr.toString());

                return encr;

                }catch(Exception e){}

                return null;

        }
    public static String decrypt(SealedObject encr){
        try{
            Cipher dece=Cipher.getInstance("RSA");
            dece.init(Cipher.DECRYPT_MODE,pair.getPublic());
            String decr=(String)encr.getObject(dece);
            System.out.println("Decryption as follows : "+decr);
            return decr;
        }catch(Exception e){}
        return null;
}
    public static void main(String[] args){
    try{
        gen=KeyPairGenerator.getInstance("RSA");
        pair=gen.generateKeyPair();
    }catch(Exception e){}
        String message;
        Scanner s=new Scanner(System.in);
        System.out.print("\n\n\nEnter the message to be hashed: ");
        message=s.nextLine();
        String md5=getMD5(message);
        SealedObject obj=encrypt(message);
        String decr=decrypt(obj);
        String md5decr=getMD5(decr);
        System.out.println("Md5: "+md5);
            System.out.println("Decrmd5: "+md5decr);
        if(md5.equals(md5decr))
```

```
                              System.out.println("Hash codes match\n\n\n");
                    else
                              System.out.println("Hash codes do not match\n\n\n");
          }
}
```

```
//digital signature using rsa
#include<bits/stdc++.h>
```

```
#define A 54059

#define B 76963

#define C 86969

using namespace std;

unsigned hash_str(const char* s)

{

   unsigned h = 31 /* also prime */;

   while (*s) {

     h = (h * A) ^ (s[0] * B);

     s++;

   }

   return h; // or return h % C;

}

bool prime(long int p)

{

    long int i=2;

    for(i=2;i<p;i++)

    if(p%i==0)

    return false;

    return true;

}


int main()

{

  char s[100];

  int i,j,k,p,q,n,phi,x,y,e,d,l,m;
```

```
 cout<<"enter the data\n";

 cin>>s;

 k=hash_str(s);

 cout<<"enter the prime numbers\n";

 cin>>p>>q;

 n=p*q;

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

 for( i=2;i<phi;i++)

  {

    if(i!=p&&i!=q&&prime(i))

    {

      for(j=2;j<phi;j++)

      if((i*j)% phi==1)

      {

        e=i;

        d=j;

        i=phi+1;

        j=phi+1;

      }

    }

  }

// d is public key n e is the private key

cout<<"encrypting the digest with d private key\n";

x=pow(k,e);

m=x%n;

cout<<"decrypting";
```

```
y=pow(x,d);

l=y%n;

if(l==k)

cout<<"the message verified";

}
```