

## EX - 1 : CaesarCipher.java

```
import java.util.Scanner;

class CaesarCipher {
    static char[] enc(String msg,int key){
        char[] crypt=new char[msg.length()];
        for(int i=0;i<msg.length();i++)
        {
            if (msg.charAt(i)==' '){
                crypt[i]=' ';
                continue;
            }
            if(msg.charAt(i)+key>122)
                crypt[i]=(char)(96+key%26);
            else
                crypt[i]=(char)(msg.charAt(i)+key);
        }
        return crypt;
    }
    public static void main(String[] args) {
        Scanner scan=new Scanner(System.in);
        String words[] = {"hey","hello","hi"};
        System.out.println("Enter the message (lower Case, without spaces): ");
        String msg=scan.nextLine();
        char[] crypt=new char[msg.length()];

        System.out.println("Enter the key value (displacement): ");
        int key=scan.nextInt();
        System.out.println("ENCRYPTED :");
        char[] encrypted = enc(msg,key);
        System.out.println(encrypted);

        System.out.println("\nDECRYPTED : ");
        char[] decrypted = enc(new String(encrypted),-1* key);
        System.out.println(decrypted);

        Scanner scan2=new Scanner(System.in);
        System.out.println("Enter the encrypted string");
        String msg2=scan2.nextLine();
        String message_parts[] = msg2.split(" ");
        int final_key = 0;
        boolean flag = false;
        for (int i=0;i<message_parts.length;i++){
            for (int j=1;j<27;j++){
                String dec_word = new String(enc(message_parts[i],-1*j));
                for (int k=0;k<words.length;k++){
                    if (dec_word.matches(words[k])){
                        System.out.println("Key Matched :"+j);
                        final_key = j;
                        flag = true;
                    }
                }
                if (flag)
                    break;
            }
            if (flag)
                break;
        }
        if (flag){
            String decrypted_message = new String(enc(msg2,-1*final_key));
            System.out.println("DECRYPTED :"+decrypted_message);
        }
    }
}
```

**SAMPLE I/O:**

Enter the message (lower Case, without spaces):

hello world

Enter the key value (displacement):

3

ENCRYPTED :

khoor zruog

DECRYPTED :

hello world

Enter the encrypted string

khoor zruog

Key Matched : 3

DECRYPTED :hello world

## EX - 1 : PlayFair.java

```
import java.io.*;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
class PlayFair{
    public static void main(String[] args){
        Scanner s=new Scanner(System.in);
        int k=0, keylen = 0, i = 0, j = 0;
        char ch;
        String key;
        char[][] mat=new char[5][5];
        System.out.println("Enter key: ");
        key = s.nextLine();
        keylen = key.length();
        Map<Character, Integer> map = new HashMap<>();
        i = j = k = 0;
        for(k = 0; k < keylen; k++){
            ch = key.charAt(k);
            if(!map.containsKey(ch)){
                map.put(ch, 1);
                mat[i][j++]=ch;
                if(j==5)
                {
                    j=0;
                    i++;
                }
            }
        }
        int newi = i, newj = j;
        ch = 'A';
        for(ch = 'A'; ch <= 'Z'; ch++){
            if(!map.containsKey(ch)){
                if(ch == 'I' || ch == 'J')
                    if(map.containsKey('I') || map.containsKey('J'))
                        continue;
                map.put(ch, 1);

                if(newj == 5){
                    if(newi == 4)
                        break;
                    newi ++;
                    newj = 0;
                }
                mat[newi][newj++]=ch;
            }
        }
        for(i=0;i<5;i++){
            for(j=0;j<5;j++){
                System.out.print(mat[i][j] + " ");
            }
            System.out.println();
        }

        System.out.println("Enter message to encrypt: ");
        String message = s.nextLine(), cipherText = "";
        int msgLen = message.length(), row1, col1, row2, col2, row, col;
        char ch1, ch2;
        boolean flag1, flag2;
        for(i = 0; i < msgLen; i++){
            ch1 = message.charAt(i++);
            if(i < msgLen)
                ch2 = message.charAt(i);
            else
                ch2 = 'X';
            if(ch1 == ch2 || (ch1 == 'I' && ch2 == 'J') || (ch1 == 'J' && ch2 ==
'I')){
                ch2 = 'X';
                i--;
            }
        }
    }
}
```

```

        flag1 = flag2 = false;
        row1 = col1 = row2 = col2 = -1;
        for(row = 0; row < 5; row++){
            for(col = 0; col < 5; col++){
                if(flag1 && flag2)
                    break;
                if(mat[row][col] == ch1 || (ch1 == 'I' && mat[row][col]
== 'J')) || (ch1 == 'J' && mat[row][col] == 'I')){
                    row1 = row;
                    col1 = col;
                    flag1 = true;
                }
                else if(mat[row][col] == ch2 || (ch2 == 'I' && mat[row]
[col] == 'J')) || (ch2 == 'J' && mat[row][col] == 'I')){
                    row2 = row;
                    col2 = col;
                    flag2 = true;
                }
            }
            if(flag1 && flag2)
                break;
        }
        if(row1 == row2){
            cipherText += (char)mat[row1][(col1+1)%5] + "" + (char)mat[row2]
[(col2+1)%5];
        }
        else if(col1 == col2){
            cipherText += (char)mat[(row1+1)%5][col1] + "" +
(char)mat[(row2+1)%5][col2];
        }
        else{
            cipherText += (char)mat[row1][col2] + "" + (char)mat[row2]
[col1];
        }
    }
    System.out.println("cipherText = "+cipherText);

    int cipherLen = cipherText.length();
    String decipheredText = "";
    for(i = 0; i < cipherLen; i++){
        ch1 = cipherText.charAt(i++);
        ch2 = cipherText.charAt(i);
        flag1 = flag2 = false;
        row1 = col1 = row2 = col2 = -1;
        for(row = 0; row < 5; row++){
            for(col = 0; col < 5; col++){
                if(flag1 && flag2)
                    break;
                if(mat[row][col] == ch1 || (ch1 == 'I' && mat[row][col]
== 'J')) || (ch1 == 'J' && mat[row][col] == 'I')){
                    row1 = row;
                    col1 = col;
                    flag1 = true;
                }
                else if(mat[row][col] == ch2 || (ch2 == 'I' && mat[row]
[col] == 'J')) || (ch2 == 'J' && mat[row][col] == 'I')){
                    row2 = row;
                    col2 = col;
                    flag2 = true;
                }
            }
            if(flag1 && flag2)
                break;
        }
        if(row1 == row2){
            if(col1 - 1 < 0)
                col1 = 5;
            if(col2 - 1 < 0)
                col2 = 5;

```

```

                                decipheredText += (char)mat[row1][(col1-1)] +""+
(char)mat[row2][(col2-1)];
                                }
                                else if(col1 == col2){
                                    if(row1 - 1 < 0)
                                        row1 = 5;
                                    if(row2 - 1 < 0)
                                        row2 = 5;
                                decipheredText += (char)mat[(row1-1)][col1] +""+
(char)mat[(row2-1)][col2];
                                }
                                else{
                                    decipheredText += (char)mat[row1][col2] + ""+(char)mat[row2]
[col1];
                                }
                            }
                        System.out.println("decipheredText = "+decipheredText);
                    }
                }
            }
        }
    }
}

```

**SAMPLE I/O:**

Enter key:

HELLOWORLD

H E L O W

R D A B C

F G I K M

N P Q S T

U V X Y Z

Enter message to encrypt:

MARYHADALITTLELAMB

cipherText = ICBULRABAQQZQWLOCIAY

decipheredText = MARYHADALITXTLELAMBX

## EX2 : HillCipher.java

```
import java.io.*;
import java.util.*;
import java.lang.Math.*;

public class HillCipher {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        System.out.println("\nHILL CIPHER\n");
        System.out.println("1. Encryption\n2. Decryption\n3. Exit");
        int choice = -1;
        Methods method= new Methods();

        while(choice != 3) {
            System.out.print("\nEnter an option : ");
            choice = scanner.nextInt();
            switch(choice) {
                case 1:
                    method.Encrypt();
                    break;
                case 2:
                    method.Decrypt();
                    break;
                case 3:
                    break;
            }
        }
    }

    class Methods {

        int GCD(int m,int n){
            if(m==0)
                return n;
            return GCD(n%m,m);
        }

        boolean Invertible(int[][] A) {
            int det = 0;
            for(int i=0;i<3;i++) {
                int a = 1;
                int b = (i+1)%3;
                int partial = (A[a][b] * A[(a+1)%3][(b+1)%3]);
                partial -= (A[a][(b+1)%3] * A[(a+1)%3][b]);
                partial *= A[0][i];
                det += partial;
            }

            if(det == 0){
                System.out.println("The given key matrix is not Invertible");
                return false;
            }

            // Have to find d^-1
            // d^-1 does not exist if gcd(d,26)<> 1
            // In that case find a different key
            if(GCD(det,26) != 1){
                System.out.println("The inverse key does not exist for the given key
matrix");
                return false;
            }
            System.out.println("The given key matrix is Invertible");
            return true;
        }

        int[][] Inverse(int[][] A) {
```

```

int det = 0;
for(int i=0;i<3;i++) {
    int a = 1;
    int b = (i+1)%3;
    int partial = (A[a][b] * A[(a+1)%3][(b+1)%3]);
    partial -= (A[a][(b+1)%3] * A[(a+1)%3][b]);
    partial *= A[0][i];
    det += partial;
}

//Find determinant modulo 26
while(det<0 || det>25){
    if(det<0)
        det+=26;
    else det%=26;
}

//Find inverse determinant
int inverseDet=0;
for(int i=1;i<=25;i++) {
    if((det*i)%26 == 1){
        inverseDet = i;
        break;
    }
}

//transpose
for(int i=0;i<3;i++){
    for(int j=0;j<i;j++){
        int temp = A[i][j];
        A[i][j] = A[j][i];
        A[j][i] = temp;
    }
}

int[][] inverseMatrix = new int[3][3];

for(int i=0;i<3;i++) {
    int minorDet = 0;
    for(int j=0;j<3;j++) {
        int a = (i+1)%3;
        int b = (j+1)%3;
        minorDet = (A[a][b] * A[(a+1)%3][(b+1)%3]);
        minorDet -= (A[a][(b+1)%3] * A[(a+1)%3][b]);
        minorDet*=inverseDet; // d^-1 * Adj(A)
        inverseMatrix[i][j] = minorDet;
    }
}
return inverseMatrix;
}

int[] MatrixMultiply(int A[],int B[][]){
    int sum[] = new int[3];
    for(int i=0;i<3;i++) {
        sum[i] = 0;
        for(int j=0;j<3;j++){
            sum[i] += (A[j]*B[j][i]);
            sum[i] = sum[i]%26;
        }
    }
    return sum;
}

public void Encrypt() {
    Scanner scanner = new Scanner(System.in);
    System.out.println("\nENCRYPTION");
    System.out.print("\nEnter the plain text : ");
    String plainText = scanner.next();
    System.out.println("Enter the key matrix : ");
    int key[][] = new int[3][3];
    for(int i=0;i<3;i++)

```



```

        for(int j=0;j<3;j++)
            key[i][j] = scanner.nextInt();

    if(!Invertible(key))
        return;

    int len = plainText.length();
    String cipherText = "";
    for(int i=0;i<len;) {
        int[] pair = new int[3];
        for(int j=0;j<3;j++) {
            if(i<len)
                pair[j] = plainText.charAt(i++) - 65;
            else pair[j] = 0;
        }

        pair = MatrixMultiply(pair,key);
        for(int j=0;j<3;j++) {
            cipherText +=(char)(pair[j] + 65);
        }
    }

    System.out.println("\nThe cipher text is : " + cipherText);
}

public void Decrypt() {

    Scanner scanner = new Scanner(System.in);
    System.out.println("\nDECRYPTION");
    System.out.print("\nEnter the cipher text : ");
    String cipherText = scanner.next();
    System.out.println("Enter the key matrix : ");
    int key[][] = new int[3][3];
    for(int i=0;i<3;i++)
        for(int j=0;j<3;j++)
            key[i][j] = scanner.nextInt();

    if(!Invertible(key))
        return;

    int[][] inverseKey = new int[3][3];

    inverseKey = Inverse(key);

    int len = cipherText.length();
    String plainText = "";
    for(int i=0;i<len;) {
        int[] pair = new int[3];
        for(int j=0;j<3;j++) {
            if(i<len)
                pair[j] = cipherText.charAt(i++) - 65;
            else pair[j] = 0;
        }

        pair = MatrixMultiply(pair,inverseKey);
        for(int j=0;j<3;j++) {
            if(pair[j] >=0)
                plainText +=(char)(pair[j] + 65);
            else plainText += (char)(65 + pair[j] + 26);
        }
    }

    System.out.println("\nThe plain text is : " + plainText);
}
}

```

## **SAMPLE I/O:**

HILL CIPHER

1. Encryption
2. Decryption
3. Exit

Enter an option : 1

ENCRYPTION

Enter the plain text : HELLOWORLD

Enter the key matrix :

1 2 3

1 2 3

1 2 3

The given key matrix is not Invertible

Enter an option : 1

ENCRYPTION

Enter the plain text : HELLOWORLD

Enter the key matrix :

6 24 1

13 16 10

20 17 15

The given key matrix is Invertible

The cipher text is : CDEMENFPLSUD

Enter an option : 2

DECRYPTION

Enter the cipher text : CDEMENFPLSUD

Enter the key matrix :

6 24 1

13 16 10

20 17 15

The given key matrix is Invertible

The plain text is : HELLOWORLDAA

Enter an option : 3

## EX2 - Vigenere.java

```
import java.io.*;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
class Vigenere{
    public static char findCharacter(char array[], char ch){
        int i = 0;
        for(i = 0; i < 26; i++){
            if(array[i] == ch)
                return (char)(65 + i);
        }
        return 0;
    }
    public static void main(String[] args){
        String input, key;
        Scanner sc = new Scanner(System.in);
        int inputLen = 0, i = 0, j = 0, keyLen = 0, k = 0;
        System.out.println("Enter input message: ");
        input = sc.nextLine();
        inputLen = input.length();

        System.out.println("Enter key: ");
        key = sc.nextLine();
        keyLen = key.length();

        k = keyLen; i = 0;
        while(k < inputLen){
            key += key.charAt(i%keyLen);
            i++;
            k++;
        }
        System.out.println("Key repeated to form: "+key);
        char VigenereMatrix[][] = new char[26][26];
        for(i = 0; i < 26; i++){
            for(j = 0; j < 26; j++){
                VigenereMatrix[i][j] = (char)(65 + ((i+j)%26));
                System.out.print(VigenereMatrix[i][j] + " ");
            }
            System.out.println();
        }

        String encryptedMessage = "";
        for(i = 0; i < inputLen; i++){
            encryptedMessage += VigenereMatrix[input.charAt(i) - 'A']
[key.charAt(i) - 'A'];
        }
        System.out.println("Encrypted Message = "+encryptedMessage);

        String decryptedMessage = "";
        for(i = 0; i < inputLen; i++){
            decryptedMessage += findCharacter(VigenereMatrix[key.charAt(i) -
'A'], encryptedMessage.charAt(i));
        }
        System.out.println("Decrypted Message = "+decryptedMessage);

    }
}
```

# SAMPLE I/0

Enter input message:

HELLOWORLD

Enter key:

APPLE

Key repeated to form: APPLEAPPLE

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z  
B C D E F G H I J K L M N O P Q R S T U V W X Y Z A  
C D E F G H I J K L M N O P Q R S T U V W X Y Z A B  
D E F G H I J K L M N O P Q R S T U V W X Y Z A B C  
E F G H I J K L M N O P Q R S T U V W X Y Z A B C D  
F G H I J K L M N O P Q R S T U V W X Y Z A B C D E  
G H I J K L M N O P Q R S T U V W X Y Z A B C D E F  
H I J K L M N O P Q R S T U V W X Y Z A B C D E F G  
I J K L M N O P Q R S T U V W X Y Z A B C D E F G H  
J K L M N O P Q R S T U V W X Y Z 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 A B C D E F G H I J  
L M N O P Q R S T U V W X Y Z A B C D E F G H I J K  
M N O P Q R S T U V W X Y Z A B C D E F G H I J K L  
N O P Q R S T U V W X Y Z A B C D E F G H I J K L M  
O P Q R S T U V W X Y Z A B C D E F G H I J K L M N  
P Q R S T U V W X Y Z A B C D E F G H I J K L M N O  
Q R S T U V W X Y Z A B C D E F G H I J K L M N O P  
R S T U V W X Y Z A B C D E F G H I J K L M N O P Q  
S T U V W X Y Z A B C D E F G H I J K L M N O P Q R  
T U V W X Y Z A B C D E F G H I J K L M N O P Q R S  
U V W X Y Z A B C D E F G H I J K L M N O P Q R S T  
V W X Y Z A B C D E F G H I J K L M N O P Q R S T U  
W X Y Z A B C D E F G H I J K L M N O P Q R S T U V  
X Y Z A B C D E F G H I J K L M N O P Q R S T U V W  
Y Z A B C D E F G H I J K L M N O P Q R S T U V W X  
Z A B C D E F G H I J K L M N O P Q R S T U V W X Y

Encrypted Message = HTAWSWDGWH

Decrypted Message = HELLOWORLD

### EX3 - RailFence.java

```
import static java.lang.Math.abs;
import java.util.Scanner;

class RailFence {
    public static void main(String[] args) {
        int key = 0; int i = 0; int j = 0; int len = 0;

        Scanner reader = new Scanner(System.in);
        Scanner keyread = new Scanner(System.in);

        System.out.println("\nEnter the key: ");
        key = keyread.nextInt();

        System.out.println("\nEnter the plain text: ");
        String message = reader.nextLine();
        len = message.length();
        System.out.println(len);
        System.out.println(key);
        char matrix [][] = new char [key][len];
        for (i=0;i<len;i++){
            for (j=0;j<key;j++){
                matrix[j][i]=' ';
            }
        }
        char[] a = message.toCharArray();
        int ind;
        int tkey = key - 1;
        for (i=0;i<len;i++){
            ind = tkey - Math.abs(tkey-i%(2*tkey));
            matrix[ind][i]=a[i];
        }
        for (i=0;i<key;i++){
            for (j=0;j<len;j++){
                System.out.print(matrix[i][j]);
                System.out.print("\n");
            }
        }
        System.out.println("ENCRYPTED :");
        for (i=0;i<key;i++){
            for (j=0;j<len;j++){
                System.out.print(matrix[i][j]);
            }
        }
        System.out.println("\n\nDECRYPTION :");
        System.out.println("\nEnter the key for decryption: ");
        int dec_key = keyread.nextInt();
        tkey = dec_key - 1;
        for (i=0;i<len;i++){
            ind = tkey - Math.abs(tkey-i%(2*tkey));
            System.out.print(matrix[ind][i]);
        }
        System.out.println();
    }
}
```

## SAMPLE I/O

Enter the key:

4

Enter the plain text:

hello world

11

4

h\*\*\*\*\*w\*\*\*\*

\*e\*\*\* \*o\*\*\*

\*\*l\*o\*\*\*r\*d

\*\*\*l\*\*\*\*\*l\*

ENCRYPTED :

h\*\*\*\*\*w\*\*\*\*\*e\*\*\* \*o\*\*\*\*\*l\*o\*\*\*r\*d\*\*\*l\*\*\*\*\*l\*

DECRYPTION :

Enter the key for decryption:

4

hello world

### EX3 - RowColCipher.java

```
import java.io.*;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
class RowColCipher{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int n = 0, i = 0, j = 0, k = 0, inputLen = 0;
        String inputMsg = "";
        char ch = ' ';
        System.out.println("Enter the message: ");
        inputMsg = sc.nextLine();
        System.out.println("Enter the number of columns: ");
        n = sc.nextInt();
        inputLen = inputMsg.length();

        /*****ENCRYPTION*****/
        char Matrix[][] = new char[100][n];
        i = j = k = 0;
        for(i = 0; k < inputLen; i++){
            for(j = 0; j < n; j++){
                while(k < inputLen && (ch = inputMsg.charAt(k)) == ' '){
                    k++;
                }
                if(k < inputLen){
                    Matrix[i][j] = ch;
                    System.out.print(Matrix[i][j] + " ");
                }
                else{
                    while(j < n){
                        Matrix[i][j++] = 'x';
                        System.out.print(Matrix[i][j-1] + " ");
                    }
                }
                k++;
            }
            System.out.println();
        }

        int m = i;

        Map<Integer, Integer> keyMap = new HashMap<>();
        String cipherText = "";
        int temp, key[] = new int[n];
        System.out.println("Enter key: ");
        for(i = 0; i < n; i++){
            temp = sc.nextInt();
            key[i] = temp-1;
            keyMap.put(temp-1, i);
        }
        for(i = 0; i < n; i++){
            for(j = 0; j < m; j++){
                cipherText += Matrix[j][keyMap.get(i)];
            }
        }
        System.out.println("Cipher Text = "+cipherText);

        /*****DECRYPTION*****/
        char decipherMatrix[][] = new char[n][m];
        k = 0;
        for(i = 0; i < n; i++){
            for(j = 0; j < m; j++){
                decipherMatrix[i][j] = cipherText.charAt(k++);
                System.out.print(decipherMatrix[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

```
String decipheredText = "";
int row = 0, col = 0;
for(col = 0; col < m; col++){
    for(row = 0; row < n; row++){
        decipheredText += decipherMatrix[key[row]][col];
    }
}

System.out.println("decipheredText = "+decipheredText);
}
```



# **SAMPLE I/O**

```
Enter the message:
hello world
Enter the number of columns:
4
h e l l
o w o r
l d x x
Enter key:
2
1
3
4
Cipher Text = ewdholloxlrx
e w d
h o l
l o x
l r x
decipheredText = helloworldxx
```

#### EX4 - DES.java

```
import java.util.Arrays;
import java.util.Scanner;
import java.util.Collections;

class DES {
    static int[][] IP_matrix = new int[][] {
        {58,50,42,34,26,18,10,2},
        {60,52,44,36,28,20,12,4},
        {62,54,46,38,30,22,14,6},
        {64,56,48,40,32,24,16,8},
        {57,49,41,33,25,17,9,1},
        {59,51,43,35,27,19,11,3},
        {61,53,45,37,29,21,13,5},
        {63,55,47,39,31,23,15,7}
    };
    static int[][] IP_inv_matrix = new int[][]{
        {40,8,48,16,56,24,64,32},
        {39,7,47,15,55,23,63,31},
        {38,6,46,14,54,22,62,30},
        {37,5,45,13,53,21,61,29},
        {36,4,44,12,52,20,60,28},
        {35,3,43,11,51,19,59,27},
        {34,2,42,10,50,18,58,26},
        {33,1,41,9,49,17,57,25}
    };
    static int[][] PC1 = new int[][] {
        {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}
    };
    static int[][] PC2 = new int[][] {
        {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}
    };
    static int[][] E = new int[][] {
        {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}
    };
    static int[][] P = new int[][] {
        {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}
    };
    static int s_boxes[][][] = new int[][][] {
        {
            {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}
}
};
public static boolean[] int2bits(int input,int num_bits){
    boolean[] bits = new boolean[num_bits];
    for (int i = num_bits-1; i >= 0; i--) {
        bits[(num_bits-1)-i] = (input & (1 << i)) != 0;
    }
    return bits;
}
public static int bits2int(boolean bits[]){
    int n=0;
    int len = bits.length;
    for (int i = len-1; i >= 0; i--) {
        // System.out.println(bits[i]+" "+Math.pow(2, (len-1)-i));
        if (bits[i]){
            n += Math.pow(2, (len-1)-i);
        }
    }
    return n;
}
public static String bits2hex(boolean bits[]){
    int num_hexes = bits.length/4;
    String hex_parts = "";
    for (int i = 0; i < num_hexes; i++) {
        int temp = bits2int(Arrays.copyOfRange(bits, i*4, (i+1)*4));
        hex_parts += Integer.toHexString(temp);
    }
    return hex_parts.toUpperCase();
}
public static boolean[] hex2bits(String hexString){
    int num_hexes = hexString.length();
    boolean[] bits = new boolean [num_hexes*4];
    boolean[] temp;
    for (int i = 0; i < num_hexes; i++) {

```

```

        temp = int2bits(Integer.parseInt(String.valueOf(hexString.charAt(i)),16), 4);
        for (int j=0;j<4;j++){
            bits[i*4+j] = temp[j];
        }
    }
    return bits;
}

public static boolean[] binstr2bits(String s){
    String[] stringparts = s.split(" ");
    String reduced_s = String.join("", stringparts);
    int num_bits = reduced_s.length();
    boolean[] bits = new boolean[num_bits];
    for (int i = 0; i < num_bits; i++) {
        if (reduced_s.charAt(i) == '1'){
            bits[i] = true;
        }
        else{
            bits[i] = false;
        }
    }
    return bits;
}

public static boolean[] bits2str(boolean[] bits){
    System.out.println(bits.length);
    for (int i = 0; i < bits.length; i++) {
        if (bits[i]){
            System.out.print("1");
        }
        else{
            System.out.print("0");
        }
    }
    System.out.println();
    return bits;
}

public static boolean[] permute(boolean[] in_bits,int[][] P) {
    int P_len = P.length*P[0].length;
    boolean[] out_bits = new boolean[P_len];
    int counter = 0;
    int index;
    for (int i = 0;i<P.length;i++){
        for(int j = 0;j<P[0].length;j++){
            index = P[i][j]-1;
            out_bits[counter++] = in_bits[index];
        }
    }
    return out_bits;
}

public static boolean[] leftShift(boolean[] in_bits,int n) {
    int index;
    int in_length = in_bits.length;
    boolean[] out_bits = new boolean[in_length];
    for (int i = 0;i<in_length;i++){
        index = (i+n)%in_length;
        out_bits[i] = in_bits[index];
    }
    return out_bits;
}

public static boolean [] concat(boolean [] first, boolean [] second) {
    boolean[] result = Arrays.copyOf(first, first.length + second.length);
    System.arraycopy(second, 0, result, first.length, second.length);
    return result;
}

public static boolean[][] keyGen(String init_key) {
    boolean[] K = binstr2bits(init_key);

```

```

boolean[] K_plus = permute(K, PC1);
boolean[][] C = new boolean[17][28];
boolean[][] D = new boolean[17][28];
boolean[][] Keys = new boolean[16][48];

C[0] = Arrays.copyOfRange(K_plus, 0, 28);
D[0] = Arrays.copyOfRange(K_plus, 28, K_plus.length);

int[] shiftNums = {1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1};
for (int i=1;i<17;i++){
    C[i] = leftShift(C[i-1], shiftNums[i-1]);
    D[i] = leftShift(D[i-1], shiftNums[i-1]);
    Keys[i-1] = permute(concat(C[i], D[i]), PC2);
}
return Keys;
}

public static boolean[] arrayXOR(boolean [] a, boolean [] b) {
    boolean[] bits = new boolean[a.length];
    for (int i =0;i<a.length;i++){
        bits[i] = a[i]^ b[i];
    }
    return bits;
}

public static boolean[] feistel(boolean[] R, boolean[] K){
    boolean[] ER = permute(R, E);
    boolean[] temp = arrayXOR(ER,K);
    boolean[][] B = new boolean[8][6];
    boolean[][] SB = new boolean[8][4];
    boolean[] pre_fin = new boolean[32];
    for (int i =0;i<8;i++){
        B[i] = Arrays.copyOfRange(temp, i*6, (i+1)*6);
        SB[i] = get_SboxVal(B[i], i);
        for(int j=i*4;j<(i+1)*4;j++){
            pre_fin[j]=SB[i][j%4];
        }
    }
    boolean[] fin = permute(pre_fin, P);
    return fin;
}

public static boolean[] get_SboxVal(boolean[] bits, int n){
    int[][] chosed_S = s_boxes[n];
    int row_num = bits2int(new boolean[] {bits[0],bits[5]});
    int col_num = bits2int(new boolean[] {bits[1],bits[2],bits[3],bits[4]});
    int chosen_num = chosed_S[row_num][col_num];
    return int2bits(chosen_num, 4);
}

public static String encrypt(String hexMessage, boolean[][] keys) {
    // boolean[] M = binstr2bits(message);
    boolean[] M = hex2bits(hexMessage);
    boolean[] IP = permute(M, IP_matrix);

    boolean[][] L = new boolean[17][32];
    boolean[][] R = new boolean[17][32];

    L[0] = Arrays.copyOfRange(IP, 0, 32);
    R[0] = Arrays.copyOfRange(IP, 32, IP.length);

    for (int i =1;i<17;i++){
        L[i] = R[i-1];
        R[i] = arrayXOR(L[i-1],feistel(R[i-1],keys[i-1]));
    }
    return bits2hex(permute(concat(R[16], L[16]),IP_inv_matrix));
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter 64 bit binary key (16 hex chars): ");
    String strKey,message;
    strKey = scanner.nextLine();
    // strKey = "00010011 00110100 01010111 01111001 10011011 10111100 11011111
11110001";

```

```
boolean[][] Keys = keyGen(strKey);
System.out.println("Enter Message (16 hex chars): ");
message = scanner.nextLine();
// message = "0123456789ABCDEF";
String encrypted = encrypt(message,Keys);
System.out.println("Encrypted : "+encrypted);

// Reverse order of Keys for decryption
Collections.reverse(Arrays.asList(Keys));
String decrypted = encrypt(encrypted,Keys);
System.out.println("Decrypted : "+decrypted);
scanner.close();
    }
}
```

# **SAMPLE I/0**

Enter 64 bit binary key (16 hex chars):

00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

Enter Message (16 hex chars):

0123456789ABCDEF

Encrypted : 85E813540F0AB405

Decrypted : 0123456789ABCDEF

## EX5 - RSA.java

```
import java.util.Arrays;
import java.util.Scanner;

class RSA{
    public static int GCD(int a, int b) {
        if(b == 0)
            return a;
        return GCD(b, a%b);
    }

    public static int[][] getKeyNums(int p, int q){
        int n = p * q;
        int phi_n = (p-1)*(q-1);
        int e,d;
        for (e =5;e<phi_n;e++){
            if (GCD(e, phi_n)==1){
                break;
            }
        }
        for (d =3;d<phi_n;d++){
            if (e*d%phi_n==1){
                break;
            }
        }
        return new int[][] {{e,n},{d,n}};
    }
    public static int modmul(int n,int exp, int mod){
        int result = n%mod;
        for (int i =1;i<exp;i++){
            result = (result*n)%mod;
        }
        return result;
    }
    public static double encrypt(int[] key,int m){
        return modmul(m,key[0],key[1]);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter p: ");
        int p = scanner.nextInt();
        System.out.print("Enter q: ");
        int q = scanner.nextInt();
        System.out.print("Enter message (number less than "+p*q+"): ");
        int msg = scanner.nextInt();
        int keys[][] = getKeyNums(p,q);
        int[] pub_key = keys[0];
        int[] pri_key = keys[1];
        System.out.println("\nPublic Key :"+Arrays.toString(pub_key));
        System.out.println("Private Key :"+Arrays.toString(pri_key));

        int enc = (int)encrypt(pub_key, msg);
        System.out.println("\nEncrypted: "+enc);

        int dec = (int)encrypt(pri_key, enc);
        System.out.println("Decrypted: "+ dec);

        scanner.close();
    }
}
```



# **SAMPLE I/0**

Enter p: 17  
Enter q: 11  
Enter message (number less than 187): 88

Public Key : [7, 187]  
Private Key : [23, 187]

Encrypted: 11  
Decrypted: 88

## EX6 - DHex.java

```
import java.util.Arrays;
import java.util.Random;;
import java.util.Scanner;

class DHex{
    public static int modmul(int n,int exp, int mod){
        int result = n%mod;
        for (int i =1;i<exp;i++){
            result = (result*n)%mod;
        }
        return result;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter q: ");
        int q = scanner.nextInt();
        System.out.print("Enter alpha: ");
        int alpha = scanner.nextInt();
        // A & B agree upon q and alpha

        Random r = new Random();
        int Xa = r.nextInt(q);
        int Xb = r.nextInt(q);
        System.out.println("Xa = "+Xa+"  Xb = "+Xb);
        // A & B randomly generate secret keys

        int Ya = modmul(alpha, Xa, q);
        int Yb = modmul(alpha, Xb, q);
        System.out.println("Ya = "+Ya+"  Yb = "+Yb);
        // A and B generate public keys and share them with each other

        int Kb = modmul(Ya, Xb, q);
        int Ka = modmul(Yb, Xa, q);
        // A and B generate shared session key
        // using other's private key and its own secret key
        System.out.println("Ka = "+Ka+"  Kb = "+Kb);
        scanner.close();
    }
}
```

# **SAMPLE I/0**

Enter q: 353

Enter alpha: 3

Xa = 27    Xb = 42

Ya = 95    Yb = 306

Ka = 50    Kb = 50