

A Mini Project report on

CRYPTOGRAPHIC SYSTEMS

submitted in partial fulfillment of the course
CSE-1007: Introduction to Cryptography

By

18BCD7044

Sharaj Chandran

18BCD7008

Kathal Aditya Rajendra

18BCN7143

Sk..Saddam Hossain



School of Computer Science & Engineering
VIT-AP UNIVERSITY, INAVOLU, AMARAVATI
OCTOBER, 2019

CONTENTS

Description	Page No.
1. Frequency Attack (Mono-additive)	1
2. Frequency Attack (Multiplicative)	6
3. Frequency Attack (Mono Alphabetic substitution)	9
4. Hill Cipher	14
5. Playfair Cipher	19
6. Vigenere Cipher	26
7. Extended Euclidian Algorithm	28
8. Polynomial Algorithm Implementation	30
9. DES Key generation	33
10. S-DES Key generation	36
11. S-DES Encryption	38
12. S-DES Decryption	41
13. AES Key generation	44
14. S-AES Key generation	59
15. S-AES Encryption	65
16. S-AES Decryption	78
17. RSA Key Generation	86
18. RSA Encryption	88
19. RSA Decryption	90
20. Mini Project Encryption	92
21. Mini Project Decryption	100

Frequency Attack

1. Write a program that can perform a letter frequency attack on an additive cipher without human intervention. Your program should produce possible plaintexts in rough order of likelihood. It would be good if your user interface allowed the user to specify "give me the top 10 possible plaintexts".
2. Write a program that can perform a letter frequency attack on any monoalphabetic substitution cipher without human intervention. Your program should produce possible plaintexts in rough order of likelihood. It would be good if your user interface allowed the user to specify "give me the top 10 possible plaintexts".

1. Code:

```
public class AdditiveAttack
{
    public static void main(String agrs[])
    {
        String s =
"XLILSYWIMWRS AJSVWEPIJSVVQMPPMSRHSPPEVWMXMWASVXLQSVILYVVCFIJSVIXLIWIPPVIM
GIMZIWQSVISJJIVW";
        Frequency obj = new Frequency();
        obj.main(s);
        char[] constC = {'E','T','A','O','I','N','S','H','R','D'};
        for(int i = 0 ; i < s.length() && i<10 ; i++)
        {
            int key = mod(obj.alpha[0] - constC[i]) ;
            System.out.print("KEY : " + key + " ");
            for(int j = 0 ; j < s.length() ; j++)
            {
                int c = (s.charAt(j) - 65);
                char t = (char)(mod((c - key)) + 65);
                System.out.print(t);
            }
            System.out.println();
        }
    }

    public static int mod(int n)
    {
```

```

        if(n < 0)
            return(n+26)%26;
        else
            return n%26;
    }
}
public class Frequency
{
    double[] frequencyDistribution;
    char[] alpha;
    public void main(String s)
    {
        //Finding the unique variable
        char[] US = new char[s.length()];
        int t = 0 ;
        for(int i = 0 ; i < s.length() ; i++)
        {
            char c = s.charAt(i);
            boolean found = false;
            for(int j = 0 ; j < US.length; j++)
            {
                if(c == US[j])
                    found = true;
            }

            if(found)
                continue;
            else
            {
                US[t] = c ;
                t++;
            }
        }

        //finding frequency
        double[] numerical = new double[US.length];
        for(int i = 0 ; i < US.length ; i++)
        {
            for(int j = 0 ; j < s.length() ; j++)
            {
                if(s.charAt(j) == US[i])
                    numerical[i]++;
            }
        }

        for(double x : numerical)
            x = (x / s.length())*100;

        bubbleSort(numerical , US);
        frequencyDistribution = numerical;
        alpha = US;
    }
}

```

```

    }

    void bubbleSort(double arr[] , char array[])
    {
        int n = arr.length;
        for (int i = 0; i < n-1; i++)
            for (int j = 0; j < n-i-1; j++)
                if (arr[j] < arr[j+1])
                {
                    // swap arr[j+1] and arr[j]
                    double temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                    char t = array[j];
                    array[j] = array[j+1];
                    array[j+1] = t;
                }
    }
}

```

OUTPUT:

```

KEY : 4   THEHOUSEISNOWFORSALEFORMILLIONDOLLARSITISWORTHMOREHURRYBEFORETHESELLERRECEIVESMOREOFFERS
KEY : 15  IWTWDJHTXHCDLUDGHPATUDGGBXAAxDCSDAAPGHXIXHLDGIWBDGTWJGGNQTTUDGTIWTHTAATGGTRTXKTHBDGTDUUTGH
KEY : 8   PDADKQOAEOKSBKNOHWABKNNIEHHEKJZKHHWNOEPEOSKNPDIKNADQNNUXABKNAPDAOAHANNAYAERAIOIKNAKBANO
KEY : 20  DRORYECOSCYGPGYBCKVOPYBBWSVVSXNYVVKBCSDSCGYBDRWYBOREBBILOPYBODROCOVVOBBOMOSFOCWYBOYPPBC
KEY : 0   XLILSYWIMWRSASJSVWEPIJSVVQMPMSRHSPPEVWXMWASVXLQSVILYVVCFIJSVIXLIWIPPPIVIGIMZIWQSVISJJIVW
KEY : 21  CQNQXDBNRBWFOXABJUNOXAAVRUURXWXMUUJABRCRBFXACQVXANQDAAHKNOXANCQNBNUUNAANLNRENBVXANXOONAB
KEY : 16  HVSVCIGSWGBCKTCFGOZSTCFFAWZZWCBRCCZOFGWHWGKCFHVACFSVIFFPSTCFSHVSGSZZSFFSQSWJSGACFSCTTSFG
KEY : 1   WKHKRXVHLVQRZIRUVDOHIRUUPLOOLRQGROODUVLWLZRUWKPRUHKXUUBEHIRUHWKHVHOOHUUHFHLYHVPRUHRIIHUV
KEY : 17  GURUBHFRVFABJSBEFNYSBEEZVYVBAQBYNEFVGVFJBEGUZBERUHEELORSBERGURFRYYREERPRVIRFZBERBSSREF
KEY : 5   SGDGNTRDHMRNVENQRZKDENVQLHKKHNMCKKZQRHSHRVNQSLNQDGTQQXADENQDSGDRDKKDDQDBDHUURLNQDNEEDQR

```

2. Code:

```
public class MultiplicativeAttack
{
    public static void main(String agrs[])
    {
        String s =
"BZSZYMQSKQNYIDYFQARSDYFFCKRRKYNHYRRAFQKBKQIYFBZCYFSZMFFELSDYFSBZSQSRRSFFSWSKXSQCYFSYDD
SFQ";

        Frequency obj = new Frequency();
        obj.main(s);
        char[] constC = {'E','T','A','O','I','N','S','H','R','D'};
        for(int i = 0 ; i < s.length() && i<10 ; i++)
        {
            int key = mod(obj.alpha[0] - constC[i]) ;
            int invkey;
            if(found(key))
                invkey = inverse(26 ,key);
            else
                continue;
            System.out.print("KEY : " + key + " ");
            for(int j = 0 ; j < s.length() ; j++)
            {
                int c = (s.charAt(j) - 65);
                char t = (char)(mod(mod(c*invkey)) + 65);
                System.out.print(t);
            }
            System.out.println();
        }
    }

    public static int mod(int n)
    {
        if(n < 0)
            return(n+26)%26;
        else
            return n%26;
    }

    public static boolean found(int k)
    {
        int key[] = {1,3,5,7,9,11,15,17,19,21,23,25};
        for(int x : key)
            if(x == k)
                return true;
        return false;
    }

    public static int inverse(int a , int b)
    {
        int s1 = 1 ;
        int s2 = 0 ;
        int t1 = 0 ;
        int t2 = 1 ;
        int q = a/b;
        int r = a%b ;
        int t = 0 ;
        while(r > 0)
        {
            a = b;
            b = r; t = s1 ;
            s1 = s2; s2 = t - q*s2; t = t1 ;
        }
    }
}
```

```

        t1 = t2; t2 = t - q*t2;
        r = a %b;
        q = a /b ;
    }
    return t2;
}

}

public class Frequency
{
    double[] frequencyDistribution;
    char[] alpha;
    public void main(String s)
    {
        //Finding the unique variable
        char[] US = new char[s.length()];
        int t = 0 ;
        for(int i = 0 ; i < s.length() ; i++)
        {
            char c = s.charAt(i);
            boolean found = false;
            for(int j = 0 ; j < US.length; j++)
            {
                if(c == US[j])
                    found = true;
            }

            if(found)
                continue;
            else
            {
                US[t] = c ;
                t++;
            }
        }

        //finding frequency
        double[] numerical = new double[US.length];
        for(int i = 0 ; i < US.length ; i++)
        {
            for(int j = 0 ; j < s.length() ; j++)
            {
                if(s.charAt(j) == US[i])
                    numerical[i]++;
            }
        }

        for(double x : numerical)
            x = (x / s.length())*100;

        bubbleSort(numerical , US);
        frequencyDistribution = numerical;
    }
}

```

```

        alpha = US;
    }

    void bubbleSort(double arr[] , char array[])
    {
        int n = arr.length;
        for (int i = 0; i < n-1; i++)
            for (int j = 0; j < n-i-1; j++)
                if (arr[j] < arr[j+1])
                {
                    // swap arr[j+1] and arr[i]
                    double temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                    char t = array[j];
                    array[j] = array[j+1];
                    array[j+1] = t;
                }
    }
}

```

OUTPUT:

```

KEY : 25  ZBIBCOKIQKNCSXCVKAJIXCVVYQJJQCNTCJJAVKQZQKSCVZBYCVIBOVVWPIXCVIZBIKIJJIVVIEIQDIKYCVICXXIVK
KEY : 5   VFOFSYOCYNKMLKBYATOLKBBQCTTCKNRKTTABYCVCYMKBFVQKBQFSBBGXOLKBOVFOYOTTOBBBOUCPOYQKBOKLLOBY
KEY : 11  THEHOUSEISNOWFORSALEFORMILLIONDOLLARSITISWORTHMOREHURRYBEFORETHESELLERRECEIVESMOREOFFERS
KEY : 1   BZSZYMQSKQNYIDYFQARSDYFFCKRRKYNHYRRAFQKBKQIYFBZCYFSZMFFELSDYFSBZSQSRRSFFSWSKXSQCYSYDDSFQ
KEY : 15  HTWTMGIWSINMEVMJIAPWVMJJOSPPSMNXMPPAJISHSIEMJHTOMJWGTGJJCZWVMJWHTWIWPPWJJWYWSFWIOMJWMVVWJI

```


Mono-Alphabetic Substitution: Code:

```
import java.util.Scanner;

public class CryptoFrequencyAttack {
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter ths ciphertext");
        String s=sc.nextLine();
        s=s.toLowerCase();
        int l=s.length();
        int i;
        char c1[]=new char[l];
        char c2[]=new char[l];
        for(i=0;i<l;i++)
        {
            c2[i]=' ';
        }
        char c;
```

```

char c3[] = {'e','t','a','o','i','n','s','h','r','d','l','c','u','m','w','f','g','y','p','b','v','k','j','x','q','z'};
char c4[] = {'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'};
int max=0,k=0,a=0;
for(i=0;i<l;i++)
{
    c1[i]=s.charAt(i);
}
int n[]=new int[26];
for(i=0;i<26;i++)
    n[i]=0;
for(int j=0;j<26;j++)
{
    for(i=0;i<l;i++)
    {
        if(c1[i]=='a')
            n[0]++;
        else if(c1[i]=='b')
            n[1]++;
        else if(c1[i]=='c')
            n[2]++;
        else if(c1[i]=='d')
            n[3]++;
        else if(c1[i]=='e')
            n[4]++;
        else if(c1[i]=='f')
            n[5]++;
        else if(c1[i]=='g')
            n[6]++;
        else if(c1[i]=='h')

```

```
        n[7]++;  
    else if(c1[i]=='i')  
        n[8]++;  
    else if(c1[i]=='j')  
        n[9]++;  
    elseif(c1[i]=='k')  
        n[10]++;  
    else if(c1[i]=='l')  
        n[11]++;  
    else if(c1[i]=='m')  
        n[12]++;  
    elseif(c1[i]=='n')  
        n[13]++;  
    elseif(c1[i]=='o')  
        n[14]++;  
    elseif(c1[i]=='p')  
        n[15]++;  
    elseif(c1[i]=='q')  
        n[16]++;  
    else if(c1[i]=='r')  
        n[17]++;  
    else if(c1[i]=='s')  
        n[18]++;  
    else if(c1[i]=='t')  
        n[19]++;  
    elseif(c1[i]=='u')  
        n[20]++;  
    elseif(c1[i]=='v')  
        n[21]++;
```

```

        else if(c1[i]=='w')
            n[22]++;
        elseif(c1[i]=='x')
            n[23]++;
        elseif(c1[i]=='y')
            n[24]++;
        else if(c1[i]=='z')
            n[25]++;
    }
    for(i=0;i<26;i++)
    {
        if(max<n[i])
        {
            max=n[i];
            k=i;
        }
    }
    c=c4[k];
    for(i=0;i<l;i++)
    {
        if(c1[i]==c)
            c2[i]=c3[a];
    }
    a++;
    for(i=0;i<l;i++)
    {
        if(c1[i]==c)
            c1[i]=' ';
    }

```

```

        for(i=0;i<26;i++)
            n[i]=0;
        max=0;
    }
    for(i=0;i<l;i++)
        System.out.print(c2[i]);

    }
}

```

Output:

```

<terminated> CryptoFrequencyAttack [Java Application]
Enter the ciphertext
cnsan jbcn jbvebjn anx
plain text :
iehoe atine atrstae oedn

```

Hill Cipher

3. Create software that can encrypt and decrypt using a 3 x 3 Hill cipher as given in Question 13.

Code:

Encryption:

```
clc
clear all

message = 'HILLCIPHERWITH3X3KEYMATRIX.';
key = '18BCD7008';
Cmessage = IntialConversin(message);
Ckey = IntialConversin(key)';
Cmessage
Ckey
ciphertext = mod(Ckey*Cmessage , 37);
ciphertext
sizeC = size(ciphertext);
ciphertext = reshape(ciphertext , [1,sizeC(1)*sizeC(2)]);
ciphertext

for i = 1:length(ciphertext)
    if(ciphertext(i) >= 0 & ciphertext(i) <= 9)
        ciphertext(i) = ciphertext(i) + 48 ;
    elseif(ciphertext(i) >= 10 & ciphertext(i) <= 35)
        ciphertext(i) = ciphertext(i) + 65 - 10;
    elseif(ciphertext(i) == 36)
        ciphertext(i) = ciphertext(i) + 10;
    end
end

ciphertext
char(ciphertext)
```

Decryption:

```
clc
clear all

ciphertext = 'MUKJGXJR10FXDZOIKCAOSLKVC9T'
invkey = [15 5 12 ; 26 4 7 ; 0 0 14]

ciphertext = InitialConversin(ciphertext);

message = mod(invkey * ciphertext, 37);

sizeC = size(message);
message = reshape(message , [1,sizeC(1)*sizeC(2)]);
message
for i = 1:length(message)
    if(message(i) >= 0 & message(i) <= 9)
        message(i) = message(i) + 48 ;
    elseif(message(i) >= 10 & message(i) <= 35)
        message(i) = message(i) + 65 - 10;
    elseif(message(i) == 36)
        message(i) = message(i) + 10;
    end
end

message
char(message)
```

Converter:

```
function r = IntialConversin(m)
    numbermessage = double(m);

    for i = 1:length(numbermessage)
        if(numbermessage(i) >= 48 & numbermessage(i) <= 57)
            numbermessage(i) = numbermessage(i) - 48 ;
        elseif(numbermessage(i) >= 65 & numbermessage(i) <= 90)
            numbermessage(i) = numbermessage(i) - 65 + 10;
        elseif(numbermessage(i) == 46)
            numbermessage(i) = numbermessage(i) - 10;
        end
    end

    if(rem(length(numbermessage),3) == 1)
        numbermessage = [numbermessage 36 36];
    elseif(rem(length(numbermessage),3) == 2)
        numbermessage = [numbermessage 36];
    end

    c = length(numbermessage)/3;
    numbermessage = reshape(numbermessage,[3,c]);
    r = numbermessage;
end
```

OUTPUT:

Encryption:

Cmessage =

17	21	25	27	29	33	14	10	18
18	12	17	32	17	3	34	29	33
21	18	14	18	3	20	22	27	36

Ckey =

1	8	11
12	13	7
0	0	8

ciphertext =

22	19	19	0	13	18	10	21	12
30	16	27	15	35	20	24	20	9
20	33	1	33	24	12	28	31	29

ciphertext =

Columns 1 through 13

22	30	20	19	16	33	19	27	1	0	15
33	13									

Columns 14 through 26

35	24	18	20	12	10	24	28	21	20	31
12	9									

Column 27

29

ciphertext =

Columns 1 through 13

77	85	75	74	71	88	74	82	49	48	70
88	68									

Columns 14 through 26

90	79	73	75	67	65	79	83	76	75	86
67	57									

Column 27

84

ans =

'MUKJGXJR10FXDZOIKCAOSLKVC9T'

Decryption:

```
ciphertext =
```

```
'MUKJGXJR10FXDZOIKCAOSLKVC9T'
```

```
invkey =
```

```
15    5    12
26    4     7
 0     0    14
```

```
message =
```

```
Columns 1 through 13
```

```
17    18    21    21    12    18    25    17    14    27    32
18    29
```

```
Columns 14 through 26
```

```
17     3    33     3    20    14    34    22    10    29    27
18    33
```

```
Column 27
```

```
36
```

```
message =
```

```
Columns 1 through 13
```

```
72    73    76    76    67    73    80    72    69    82    87
73    84
```

```
Columns 14 through 26
```

```
72    51    88    51    75    69    89    77    65    84    82
73    88
```

```
Column 27
```

```
46
```

```
ans =
```

```
'HILLCIPHERWITH3X3KEYMATRIX.'
```

Playfair Cipher

4. Create a software that can encrypt and decrypt message using Playfair Cipher with specific key (You can consider any key of your choice). If you can implement for the generalized key means user can input the key at run time then more weightage will be given for such solution.

Code:

```
import java.util.Scanner;

public class Playfair1
{
    static char[][] a,b;
    static char[][] key = new char[5][];
    static String plaintext="HELLOBOBCOMESOOON";
    static String decrypt="EOZAIQLNPVLW";
    public static void main(String agrs[])
    {
        key = keyinput();
        System.out.println("Encrypting the message Hello Bob Come Soon");
        //Encryption
        a = converterM(plaintext);
        encrypt();
        display(a,plaintext);

        System.out.println();
        System.out.println("Decrypting the message cryptography");
        //Decryption
```

```

        b = converterM(decrypt);
        decrypt();
        display(b,decrypt);
    }
    public static char[][] converterM(String s)
    {
        char[] a = new char[s.length()];
        char[][] b = new char[2][s.length()];
        int y = 0 ;
        for(int i=0;i<s.length();i++)
            a[i] =s.charAt(i);
        for(int i = 0 ; i <s.length() ; y++)
        {
            if(a[i] != a[i+1])
            {
                b[0][y] = a[i];
                b[1][y] = a[i+1];
                i= i+ 2;
            }
            else
            {
                b[0][y] = a[i];
                b[1][y] = 'X';
                i++;
            }
        }
    }
}

```

```

        return b;
    }
    public static void encrypt()
    {
        for(int i = 0 ; i < plaintext.length() && a[0][i] != a[1][i] ; i++)
        {
            int dim1[] = finder(a[0][i]);
            int dim2[] = finder(a[1][i]);

            if(dim1[0] == dim2[0])
            {
                a[0][i] = key[(dim1[0])%5][(dim1[1]+1)%5];
                a[1][i] = key[(dim2[0])%5][(dim2[1]+1)%5];
            }
            else if(dim1[1] == dim2[1])
            {
                a[0][i] = key[(dim1[0]+1)%5][(dim1[1])%5];
                a[1][i] = key[(dim2[0]+1)%5][(dim2[1])%5];
            }
            else
            {
                a[0][i] = key[(dim1[0])%5][(dim2[1])%5];
                a[1][i] = key[(dim2[0])%5][(dim1[1])%5];
            }
        }
    }
}

```

```

    }
    public static int[] finder(int a)
    {
        if(a == 'J')
            a = 'I';

        int dim[] = new int[2];
        for(int i=0 ; i < 5 ; i++)
            for(int j = 0 ; j <5 ; j++)
            {
                if((int)(key[i][j]) == (int)(a))
                {
                    dim[0]=i;
                    dim[1]=j;
                }
            }

        return dim;
    }
    public static void display(char[][] c , String s)
    {
        for(int i=0 ; i < s.length() ; i++)
            for(int j = 0 ; j < 2 ; j++)
                System.out.print(c[j][i]);
    }
    public static void decrypt()
    {

```

```

for(int i = 0 ; i < plaintext.length() && b[0][i] != b[1][i] ; i++)
{
    int dim1[] = finder(b[0][i]);
    int dim2[] = finder(b[1][i]);
    if(dim1[0] == dim2[0])
    {
        b[0][i] = key[modifier((dim1[0])%5)][modifier((dim1[1]-1)%5)];
        b[1][i] = key[modifier((dim2[0])%5)][modifier((dim2[1]-1)%5)];
    }
    else if(dim1[1] == dim2[1])
    {
        b[0][i] = key[modifier((dim1[0]-1)%5)][(dim1[1])%5];
        b[1][i] = key[modifier((dim2[0]-1)%5)][(dim2[1])%5];
    }
    else
    {
        b[0][i] = key[(dim1[0])%5][(dim2[1])%5];
        b[1][i] = key[(dim2[0])%5][(dim1[1])%5];
    }
}

}

public static int modifier(int c)
{
    if(c < 0 )
        return (c+5)%5;
    else

```

```

        return c%5 ;

    }

    public static char[][] keyinput()
    {
        Scanner br = new Scanner(System.in);
        System.out.println("Enter the key for the cipher");
        String a[] = new String[5];
        for(int i = 0 ; i < 5 ; i++)
        {
            a[i] = br.next();
        }

        char b[][] = new char[5][5];
        int i = 0 ;
        for(String x : a)
        {
            b[i] = x.toCharArray();
            i++;
        }

        return b;
    }
}

```


OUTPUT:

```
Enter the key for the cipher
VITAP
BCDEF
GHKLM
NOQRS
UWXYZ
Encrypting the message Hello Bob Come Soon
LCKYHRCNCD SHFRQWQO
Decrypting the message cryptography
CRYPTOGRAPHY |
```

Vigenere Cipher

5. Write a program to implement Vigenere Cipher. You have to write a method for both encryption and decryption of the message.

Code:

```
import java.util.*;
public class Vignere
{
    static int[] key = {15,00,18,02,00,11};
    public static void main(String args[])
    {
        int[] message = {18,7,4,8,18,11,8,18,19,4,13,8,13,6};
        encryption(message);
        System.out.println();
        int[] cipher = {7,7,22,10,18,22,23,18,11,6,13,19,2,6};
        decryption(cipher);
    }

    public static void encryption(int[] message)
    {
        int[] cipher = new int[message.length];
        for(int i = 0 ; i < message.length ; i++)
        {
            cipher[i] = message[i] + key[i%key.length];
            cipher[i] = cipher[i]%26;
        }

        lettermaker(cipher);
    }
}
```

```

public static void decryption(int[] cipher)
{
    int[] message = new int[cipher.length];

    for(int i = 0 ; i < cipher.length ; i++)
    {
        message[i] = cipher[i] - key[i%key.length];

        if(message[i] < 0)
            message[i] = message[i] + 26 ;

        message[i] = message[i]%26;
    }

    lettermaker(message);
}

public static int[] moduli(int[] c)
{
    for(int x : c)
    {
        if(x < 0)
            x = x + 25;

        x = x%25 ;
    }

    return c;
}

public static void lettermaker(int[] a)
{
    for(int x : a)
    {
        char y = (char)(x+65);
        System.out.print(y);
    }
}
}

```

HHWKSXSLGNTCG
SHEISLISTENING

Extended Euclidian Algorithm

6. Write a program for finding multiplicative inverse using Extended Euclidean Algorithm.

Code:

```
import java.util.*;

public class Extend_Elucidean_Algo
{
    static Scanner br = new Scanner(System.in);

    public static void main(String a[])
    {
        System.out.println("Enter the first variable");
        int x = br.nextInt();
        System.out.println("Enter the second variable");
        int y = br.nextInt();

        gcd(Math.abs(x), Math.abs(y));
    }

    public static void gcd(int a , int b)
    {
        int s1 = 1 ;
        int s2 = 0 ;
        int t1 = 0 ;
        int t2 = 1 ;
        int q = a/b;
        int ad = a;
        int bd = b ;
        int r = a%b ;

        int t = 0 ;

        while(r > 0)
        {
            a = b;
            b = r;
            t = s1 ;
            s1 = s2;
            s2 = t - q*s2;
            t = t1 ;
            t1 = t2;
            t2 = t - q*t2;

            System.out.println(s2 + " " + t2+ " "+q);
            r = a %b;
            q = a /b ;

            System.out.println(t1 + " "+ t2 + " " + s1 + " "+ s2);
        }

        System.out.println("GCD: " + b);
        System.out.println(s2*ad + t2*bd == b);
    }
}
```

```
}
```

OUTPUT:

```
Enter the first variable
```

```
161
```

```
Enter the second variable
```

```
28
```

```
| 1 -5 5
```

```
1 -5 0 1
```

```
-1 6 1
```

```
-5 6 1 -1
```

```
GCD: 7
```

```
true
```

Polynomial Multiplication by computer algorithm implementation

7. Write a program to compute the multiplication of two polynomial equation in $GF(2^8)$.

Consider the given irreducible polynomial equation is $x^8 + x^4 + x^3 + x + 1$.

Code:

```
import java.util.Scanner;
public class PolynomialCrypto {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int[] n1= {0,0,0,1,1,0,1,1};
        int l=n1.length;
        int n2[]=new int[l+1];
        int n3[]=new int[l];
        int n4[][]=new int[l][l];
        int n5[]= {0,0,0,0,0,0,0,0};
        int n6[]=new int[l];
        int i,j,k=0;
        System.out.println("Enter the 1st equation");
        for(i=0;i<l+1;i++)
            n2[i]=sc.nextInt();
        System.out.println("Enter the 2nd equation");
        for(i=0;i<l;i++)
            n3[i]=sc.nextInt();
        int n=l;
        for(i=0;i<l;i++)
        {
            if(n2[i]==1)
                break;
            n--;
        }
        for(i=0;i<l;i++)
            n4[0][i]=n3[i];
        for(i=1;i<n+1;i++)
        {
            n3=polyShift(n3);
            if(n4[i-1][0]==1)
                n3=polyAdd(n1,n3);
            for(j=0;j<l;j++)
                n4[i][j]=n3[j];
        }
        for(i=0;i<l+1;i++)
        {
            if(n2[l-i]==1)
            {
                k++;
                if(k==1)
                {
```

```

                for(j=0;j<l;j++)
                {
                    n5[j]=n4[i][j];
                }
            }
            else
            {
                for(j=0;j<l;j++)
                {
                    n6[j]=n4[i][j];
                }
                n5=polyAdd(n5,n6);
            }
        }
    }
    System.out.println("Final Answer: ");
    for(i=0;i<l;i++)
        System.out.print(n5[i]);
}
static int[] polyAdd(int a[],int b[])
{
    int[] c=new int[a.length];
    for(int i=0;i<a.length;i++)
    {
        if(a[i]==0&& b[i]==0)
            c[i]=0;
        else if(a[i]==1&& b[i]==0)
            c[i]=1;
        else if(a[i]==0&& b[i]==1)
            c[i]=1;
        else
            c[i]=0;
    }
    return c;
}
static int[] polyShift(int a[])
{
    int[] b=new int[a.length];
    for(int i=0;i<a.length-1;i++)
        b[i]=a[i+1];
    b[a.length-1]=0;
    return b;
}
}

```

Output:

Enter the 1st equation

0 0 0 1 0 0 1 1 0

Enter the 2nd equation

1 0 0 1 1 1 1 0

Final Answer:

00101111

DES Keygen:

CODE:

```
public class OriginalKeyGen
{
    static int[] key = {1,0,1,0,1,0,1,0,1,0,1,1,1,0,1,1,0,0,0,0,1,0,0,1,0,0,0,1,
                        1,0,0,0,
                        0,0,1,0,
                        0,1,1,1,
                        0,0,1,1,
                        0,1,1,0,
                        1,1,0,0,
                        1,1,0,0,
                        1,1,0,1,
                        1,1,0,1};

    static int[] reducedkey = new int[56];
    static int[] left = new int[28]; static
    int[] right = new int[28]; public
    static void paritydropper()
    {
        int[] p =
        {57,49,41,33,25,17,9,1,58,50,42,34,26,18,10,2,59,51,43,35,27,19,11,3,60,52,44,36,63,5
        5,47,39,31,23,15,7,62,54,46,38,30,22,14,6,61,53,45,37,29,21,13,5,28,20,12,4};
        int a[] = new int[p.length];
        for(int i = 0 ; i<p.length ; i++)
        {
            a[i] = key[p[i]-1];
        }

        reducedkey = a;
        for(int x : a)
            System.out.print(x);
        System.out.println();
    }
    public static void separator()
    {
        for(int i = 0 ; i < 28 ; i++)
        {
            left[i] = reducedkey[i];
            right[i] = reducedkey[i+28];
        }
    }
    public static int[] leftshifter(int[] a ,int n)
    {
        if(n == 1 || n == 2 || n == 9 || n == 16)
        {
            int b = a[0];
            for(int i = 0 ; i <27 ; i++)
                a[i] = a[i+1];

            a[a.length-1] = b ;
            return a;
        }
    }
}
```

```

    }
    else
    {
        int b0 =a[0];
        int b1 =a[1];
        for(int i = 0 ; i < 26 ; i++)
            a[i] = a[i+2];
        a[26] = b0;
        a[27] = b1;
        return a;
    }
}

public static void combine()
{
    for(int i = 0 ; i < 28 ; i++)
    {
        reducedkey[i] = left[i];
        reducedkey[i+28] = right[i];
    }
}

public static int[] compressor()
{
    int[] p =
{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[] a = new int[p.length];
    for(int i = 0 ; i < p.length ; i++)
    {
        a[i] = reducedkey[p[i]-1];
    }

    return a;
}

public static void display(int[][] a , int i)
{
    for(int j = 0 ; j<48 ; j++)
    {
        System.out.print(a[i-1][j]);
    }

    System.out.println();
}

public static void main(String agrs[])
{
    int[][] keyfinal = new int[16][] ;

    paritydropper();
    separator();

    for(int i = 1 ; i <= 16 ; i++)
    {
        right = leftshifter(right,i);
        left = leftshifter(left,i);
        combine();
        keyfinal[i-1] = compressor();
    }
}

```

```

        display(keyfinal,i);
    }
}

```

OUTPUT:

```

11000011110000000011001110100011001111110000110011111010
000110010100110011010000011100101101111010001100
010001010110100001011000000110101011110011001110
000001101110110110100100101011001111010110110101
110110100010110100000011001010110110111011100011
011010011010011000101001111111101100100100010011
110000011001010010001110100001110100011101011110
011100001000101011010010110111011001111000000
001101001111100000100010111100001100011001101101
100001001011101101000100011100111101110011001100
000000100111011001010111000010001011010110111111
011011010101010101100000101011110111110010100101
110000101100000111101001011010100100101111110011
100110011100001100010011100101111100100100011111
001001010001101110001011110001110001011111010000
001100110011000011000101110110011010001101101101
000110000001110001011101011101011100011001101101

```

SDES Keygen:

Code:

```
public class KeyGen extends Thread
{
    public static int[] key = {0,0,1,0,0,1,0,1,1,1};
    public static int[] right = new int[5];
    public static int[] left = new int[5];
    public static int[] keyone;
    public static int[] keytwo;
    public static int[][] keyfinal = {keyone , keytwo};
    public static void shifter(int[] a , int n)
    {
        for(int i = 1 ; i <= n ; i++)
        {
            int b = a[0];
            for(int j = 0 ; j<4 ; j++)
                a[j] = a[j+1];

            a[4] = b ;
        }
    }
    public static int[] compression()
    {
        int[] b = {5,2,7,3,6,4,9,8};
        int[] a = new int[b.length];
        for(int i = 0 ; i < b.length ; i++)
        {
            a[i] = key[b[i]];
        }

        return a ;
    }
    public static void separator()
    {
        for(int i = 0;i<5;i++)
        {
            right[i] = key[i];
            left[i] = key[i+5];
        }
    }
    public static void combine()
    {
        for(int i = 0 ; i < 5 ; i++)
        {
            key[i] = right[i];
            key[i+5] = left[i];
        }
    }
    public static void permutation()
    {
    }
```

```

    int[] p = {2,4,1,6,3,9,0,8,7,5};
    int[] a = new int[10];
    for(int i = 0 ; i< p.length ; i++)
    {
        a[i] = key[p[i]];
    }

    key = a;
}
public static void main(String agrs[])
{
    permutation();
    separator();
    shifter(right , 1);
    shifter(left,1);
    combine();
    keyone = compression();
    for(int x : keyone)
        System.out.print(x);
    System.out.println();
    shifter(right,2);
    shifter(left,2);
    combine();
    keytwo = compression();
    for(int x : keytwo)
        System.out.print(x);
}
}

```

Output:

```

00101111
11101010

```

SDES Encryption:

Code:

```
public class Encryption
{
    static int[] plain = {1,0,1,0,0,1,0,1};
    static int[] right = new int[4];
    static int[] left = new int[4];
    static int[] cipher = new int[8];
    static int[][] keyfinal = {{0,0,1,0,1,1,1,1},{1,1,1,0,1,0,1,0}};
    public static void permutation()
    {
        int[] p = {1,5,2,0,3,7,4,6};
        int[] a = new int[8];
        for(int i = 0 ; i< p.length ; i++)
        {
            a[i] = plain[p[i]];
        }

        plain = a;
    }
    public static void separator()
    {
        for(int i = 0;i<4;i++)
        {
            left[i] = plain[i];
            right[i] = plain[i+4];
        }
    }
    public static int[] expansion(int[]c)
    {
        int[] p = {3,0,1,2,1,2,3,0};
        int[] b = new int[8];
        for(int i = 0 ; i< p.length ; i++)
        {
            b[i] = c[p[i]];
        }

        return b;
    }
    public static void swapper()
    {
        int[] a = right;
        right = left;
        left = a;
    }
    public static int[] permutationfour(int[] a)
    {
        int[] p = {1,3,2,0};
        int[] b = new int[4];
        for(int i = 0 ; i< p.length ; i++)
        {
            b[i] = a[p[i]];
        }
    }
}
```

```

        b[i] = a[p[i]];
    }
    return b;
}
public static int[] sbbox(int[] a)
{
    int[][] szero = {{1,0,3,2},{3,2,1,0},{0,2,1,3},{3,1,3,2}};
    int[][] sone = {{0,1,2,3},{2,0,1,3},{3,0,1,0},{2,1,0,3}};
    int one,two,three,four;
    one = a[0]*2 + a[3];
    two = a[1]*2 + a[2];
    three = a[4]*2 + a[7];
    four = a[5]*2 + a[6]; int
    o = szero[one][two];
    int t = sone[three][four];
    int[] b = {o/2,o%2,t/2,t%2};
    return b;
}
public static int[] function(int n)
{
    int[] a = expansion(right);
    for(int i = 0 ; i < 8 ; i++)
        a[i] = keyfinal[n][i]^a[i] ;
    a = sbbox(a);
    a = permutationfour(a);
    for(int i = 0 ; i < a.length ; i++)
        a[i] = a[i]^left[i];
    return a;
}
public static void combine()
{
    for(int i = 0 ; i < 4 ; i++)
    {
        plain[i] = left[i];
        plain[i+4] = right[i];
    }
}
public static void ipermutation()
{
    int[] p = {3,0,2,4,6,1,7,5};
    int[] a = new int[8];
    for(int i = 0 ; i < p.length ; i++)
    {
        a[i] = plain[p[i]];
    }
    cipher = a;
}
public static void main(String agrs[]) throws InterruptedException
{
    permutation();
    separator();
    left = function(0);
    swapper();
    left = function(1);
}

```

```
        combine();  
        ipermutation();  
        System.out.println("Ciphertext is:");  
        for(int x: cipher)  
            System.out.print(x);  
    }  
}
```

OUTPUT:

```
Ciphertext is:  
00110110
```


SDES Decryption:

Code:

```
public class Decryption
{
    static int[] plain = {0,0,1,1,0,1,1,0};
    static int[] right = new int[4];
    static int[] left = new int[4];
    static int[] cipher = new int[8];
    static int[][] keyfinal = {{1,1,1,0,1,0,1,0},{0,0,1,0,1,1,1,1}};
    public static void permutation()
    {
        int[] p = {1,5,2,0,3,7,4,6};
        int[] a = new int[8];
        for(int i = 0 ; i< p.length ; i++)
        {
            a[i] = plain[p[i]];
        }

        plain = a;
    }
    public static void separator()
    {
        for(int i = 0;i<4;i++)
        {
            left[i] = plain[i];
            right[i] = plain[i+4];
        }
    }
    public static int[] expansion(int[]c)
    {
        int[] p = {3,0,1,2,1,2,3,0};
        int[] b = new int[8];
        for(int i = 0 ; i< p.length ; i++)
        {
            b[i] = c[p[i]];
        }

        return b;
    }
    public static void swapper()
    {
        int[] a = right;
        right = left;
        left = a;
    }
    public static int[] permutationfour(int[] a)
    {
        int[] p = {1,3,2,0};
        int[] b = new int[4];
        for(int i = 0 ; i< p.length ; i++)
        {
```

```

        b[i] = a[p[i]];
    }
    return b;
}
public static int[] sbbox(int[] a)
{
    int[][] szero = {{1,0,3,2},{3,2,1,0},{0,2,1,3},{3,1,3,2}};
    int[][] sone = {{0,1,2,3},{2,0,1,3},{3,0,1,0},{2,1,0,3}};
    int one,two,three,four;
    one = a[0]*2 + a[3];
    two = a[1]*2 + a[2];
    three = a[4]*2 + a[7];
    four = a[5]*2 + a[6];
    int o = szero[one][two];
    int t = sone[three][four];
    int[] b = {o/2,o%2,t/2,t%2};
    return b;
}
public static int[] function(int n)
{
    int[] a = expansion(right);
    for(int i = 0 ; i < 8 ; i++)
        a[i] = keyfinal[n][i]^a[i] ;
    a = sbbox(a);
    a = permutationfour(a);
    for(int i = 0 ; i < a.length ; i++)
        a[i] = a[i]^left[i];
    return a;
}
public static void combine()
{
    for(int i = 0 ; i < 4 ; i++)
    {
        plain[i] = left[i];
        plain[i+4] = right[i];
    }
}
public static void ipermutation()
{
    int[] p = {3,0,2,4,6,1,7,5};
    int[] a = new int[8];
    for(int i = 0 ; i < p.length ; i++)
    {
        a[i] = plain[p[i]];
    }
    cipher = a;
}
public static void main(String args[]) throws InterruptedException
{
    permutation();
    separator();
    left = function(0);
    swapper();
    left = function(1);
}

```

```
        combine();  
        ipermutation();  
        System.out.println("Ciphertext is:");  
        for(int x: cipher)  
            System.out.print(x);  
    }  
}
```

Output:

```
Plaintext is:  
10100101
```

AES Keygen:

CODE:

```
import java.util.Scanner;

public class AES {
    static int z;

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int j;
        System.out.println("Enter the key");
        String s=sc.next();
        char k[]=new char[s.length()];
        char ktemp[]=new char[s.length()];
```

```

        for(j=0;j<s.length();j++)
            k[j]=s.charAt(j);
        for(z=0;z<10;z++)
        {
            ktemp=keyGen(k);
            System.out.print("Round "+(z+1)+" : ");
            System.out.print(ktemp);
            System.out.println();
            k=ktemp;
        }
    }
    static char[] keyGen(char[] k)
    {
        char[] a=new char[k.length/4];
        char[] a1=new char[k.length/4];
        char[] a2=new char[k.length/4];
        char[] a3=new char[k.length/4];
        char[] a4=new char[k.length];
        int i;
        char [] w0=new char[k.length/4];
        char [] w1=new char[k.length/4];
        char [] w2=new char[k.length/4];
        char [] w3 =newchar[k.length/4];
        char [] w4 =newchar[k.length/4];
        for(i=0;i<8;i++)
        {
            w0[i]=k[i];

```

```
}  
for(i=0;i<8;i++)  
{  
    w1[i]=k[8+i];  
}  
for(i=0;i<8;i++)  
{  
    w2[i]=k[16+i];  
}  
for(i=0;i<8;i++)  
{  
    w3[i]=k[24+i];  
}  
for(i=0;i<8;i++)  
    w4[i]=w3[i];  
char c1=w3[0];  
char c2=w3[1];  
w3[0]=w3[2];  
w3[1]=w3[3];  
w3[2]=w3[4];  
w3[3]=w3[5];  
w3[4]=w3[6];  
w3[5]=w3[7];  
w3[6]=c1;  
w3[7]=c2;  
w3=subWord(w3);  
w3=xor(w3,rConst());
```

```

        a=xor(w0,w3);
        a1=xor(a,w1);
        a2=xor(a1,w2);
        a3=xor(a2,w4);
        for(i=0;i<8;i++)
        {
            a4[i]=a[i];
        }
        for(i=0;i<8;i++)
        {
            a4[i+8]=a1[i];
        }
        for(i=0;i<8;i++)
        {
            a4[i+16]=a2[i];
        }
        for(i=0;i<8;i++)
        {
            a4[i+24]=a3[i];
        }
        return a4;
    }
    static char[] binToHexa(int k[])
    {
        int l=k.length;
        char[] a=new char[l/4];
        String s="";

```

```
for(int i=0;i<l/4;i++)
{
    if(k[4*i]==0)
        s=s+"0";
    else
        s=s+"1";
    if(k[4*i+1]==0)
        s=s+"0";
    else
        s=s+"1";
    if(k[4*i+2]==0)
        s=s+"0";
    else
        s=s+"1";
    if(k[4*i+3]==0)
        s=s+"0";
    else
        s=s+"1";
    if(s.equals("0000"))
        a[i]='0';
    else if(s.equals("0001"))
        a[i]='1';
    else if(s.equals("0010"))
        a[i]='2';
    else if(s.equals("0011"))
        a[i]='3';
    else if(s.equals("0100"))
```



```
        a[i]='4';
    else if(s.equals("0101"))
        a[i]='5';
    else if(s.equals("0110"))
        a[i]='6';
    else if(s.equals("0111"))
        a[i]='7';
    else if(s.equals("1000"))
        a[i]='8';
    else if(s.equals("1001"))
        a[i]='9';
    else if(s.equals("1010"))
        a[i]='A';
    else if(s.equals("1011"))
        a[i]='B';
    else if(s.equals("1100"))
        a[i]='C';
    else if(s.equals("1101"))
        a[i]='D';
    else if(s.equals("1110"))
        a[i]='E';
    else
        a[i]='F';
    s="";
}
return(a);
}
```

```
static int[] hexaToBin(char k[])
{
    int l=k.length;
    int[] a=new int[4*l];
    int i=0,j;
    char c;
    for(i=0;i<l;i++)
    {
        c=k[i];
        if(c=='0')
        {
            a[4*i]=0;
            a[4*i+1]=0;
            a[4*i+2]=0;
            a[4*i+3]=0;
        }
        else if(c=='1')
        {
            a[4*i]=0;
            a[4*i+1]=0;
            a[4*i+2]=0;
            a[4*i+3]=1;
        }
        else if(c=='2')
        {
            a[4*i]=0;
            a[4*i+1]=0;
```

```
        a[4*i+2]=1;
        a[4*i+3]=0;
    }
    else if(c=='3')
    {
        a[4*i]=0;
        a[4*i+1]=0;
        a[4*i+2]=1;
        a[4*i+3]=1;
    }
    else if(c=='4')
    {
        a[4*i]=0;
        a[4*i+1]=1;
        a[4*i+2]=0;
        a[4*i+3]=0;
    }
    else if(c=='5')
    {
        a[4*i]=0;
        a[4*i+1]=1;
        a[4*i+2]=0;
        a[4*i+3]=1;
    }
    else if(c=='6')
    {
        a[4*i]=0;
```

```
        a[4*i+1]=1;
        a[4*i+2]=1;
        a[4*i+3]=0;
    }
    else if(c=='7')
    {
        a[4*i]=0;
        a[4*i+1]=1;
        a[4*i+2]=1;
        a[4*i+3]=1;
    }
    else if(c=='8')
    {
        a[4*i]=1;
        a[4*i+1]=0;
        a[4*i+2]=0;
        a[4*i+3]=0;
    }
    else if(c=='9')
    {
        a[4*i]=1;
        a[4*i+1]=0;
        a[4*i+2]=0;
        a[4*i+3]=1;
    }
    else if(c=='A')
    {
```

```
        a[4*i]=1;
        a[4*i+1]=0;
        a[4*i+2]=1;
        a[4*i+3]=0;
    }
    else if(c=='B')
    {
        a[4*i]=1;
        a[4*i+1]=0;
        a[4*i+2]=1;
        a[4*i+3]=1;
    }
    else if(c=='C')
    {
        a[4*i]=1;
        a[4*i+1]=1;
        a[4*i+2]=0;
        a[4*i+3]=0;
    }
    else if(c=='D')
    {
        a[4*i]=1;
        a[4*i+1]=1;
        a[4*i+2]=0;
        a[4*i+3]=1;
    }
    else if(c=='E')
```

```

        {
            a[4*i]=1;
            a[4*i+1]=1;
            a[4*i+2]=1;
            a[4*i+3]=0;
        }
        else
        {
            a[4*i]=1;
            a[4*i+1]=1;
            a[4*i+2]=1;
            a[4*i+3]=1;
        }
    }
    return(a);
}

```

```

static char[] xor(char[] k1,char[] k2)
{
    int[] a1=hexaToBin(k1);
    int[] a2=hexaToBin(k2);
    int[] a3=new int[a1.length];
    for(int i=0;i<a1.length;i++)
    {
        if(a1[i]==0&&a2[i]==0)
            a3[i]=0;
        else if(a1[i]==1&&a2[i]==0)
            a3[i]=1;
    }
}

```

```

        else if(a1[i]==0&&a2[i]==1)
            a3[i]=1;
        else
            a3[i]=0;
    }

    char a[]=binToHexa(a3);
    return a;
}

static char[] rConst()
{
    char[] a1= {'0','1','0','0','0','0','0','0'};
    char[] a2= {'0','2','0','0','0','0','0','0'};
    char[] a3= {'0','4','0','0','0','0','0','0'};
    char[] a4= {'0','8','0','0','0','0','0','0'};
    char[] a5= {'1','0','0','0','0','0','0','0'};
    char[] a6= {'2','0','0','0','0','0','0','0'};
    char[] a7= {'4','0','0','0','0','0','0','0'};
    char[] a8= {'8','0','0','0','0','0','0','0'};
    char[] a9= {'1','B','0','0','0','0','0','0'};
    char[] a10={'3','6','0','0','0','0','0','0'};
    if(z==0)
        return a1;
    else if(z==1)
        return a2;
    else if(z==2)
        return a3;
    else if(z==3)

```

```

        return a4;
    else if(z==4)
        return a5;
    else if(z==5)
        return a6;
    else if(z==6)
        return a7;
    else if(z==7)
        return a8;
    else if(z==8)
        return a9;
    else
        return a10;
    }

    static char[] subWord(char[] c)
    {
        char[] a=new char[c.length];

        String b[][]=
        {"63","7C","77","7B","F2","6B","6F","C5","30","01","67","2B","FE","D7","AB","76"},

        {"CA","82","C9","7D","FA","59","47","F0","AD","D4","A2","AF","9C","A4","72","C0"},

        {"B7","FD","93","26","36","3F","F7","CC","34","A5","E5","F1","71","D8","31","15"},

        {"04","C7","23","C3","18","96","05","9A","07","12","80","E2","EB","27","B2","75"},

        {"09","83","2C","1A","1B","6E","5A","A0","52","3B","D6","B3","29","E3","2F","84"},

        {"53","D1","00","ED","20","FC","B1","5B","6A","CB","BE","39","4A","4C","58","CF"},

```



```

{"D0","EF","AA","FB","43","4D","33","85","45","F9","02","7F","50","3C","9F","AB"},

{"51","A3","40","8F","92","9D","38","F5","BC","B6","DA","21","10","FF","F3","D2"},

{"CD","0C","13","EC","5F","97","44","17","C4","A7","7E","3D","64","5D","19","73"},

{"60","81","4F","DC","22","2A","90","88","46","EE","B8","14","DE","5E","0B","DB"},

{"E0","32","3A","0A","49","06","24","5C","C2","D3","AC","62","91","95","E4","79"},

{"E7","CB","37","6D","8D","D5","4E","A9","6C","56","F4","EA","65","7A","AE","08"},

{"BA","78","25","2E","1C","A6","B4","C6","E8","DD","74","1F","4B","BD","8B","8A"},

{"70","3E","B5","66","48","03","F6","0E","61","35","57","B9","86","C1","1D","9E"},

{"E1","F8","98","11","69","D9","8E","94","9B","1E","87","E9","CE","55","28","DF"},

{"8C","A1","89","0D","BF","E6","42","68","41","99","2D","0F","B0","54","BB","16"};

    char[] b1= {'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
    int i,j,k1=0,k2=0;
    for(i=0;i<a.length;i=i+2)
    {
        for(j=0;j<b1.length;j++)
        {
            if(c[i]==b1[j])
                k1=j;
        }
        for(j=0;j<b1.length;j++)

```

```

        {
            if(c[i+1]==b1[j])
                k2=j;
        }
        a[i]=b[k1][k2].charAt(0);
        a[i+1]=b[k1][k2].charAt(1);
    }
    return a;
}
}

```

OUTPUT:

```

Enter the key
2475A2B33475568831E2120013AA5487
Round 1 : 8955B5CEBD20E3468CC2F1469F68A5C1
Round 2 : CE53CD1573732E53FFB1DF1560D97AD4
Round 3 : FF8985C58CFAAB96734B748313920E57
Round 4 : B822DEB834D8752E479301AD54010FFA
Round 5 : D454F398E08C86B6A71F871BF31E88E1
Round 6 : 86900B95661C8D23C1030A38321D82D9
Round 7 : 62833EB6049FB395C59CB9ADF7813B74
Round 8 : EE61ACDEEAFE1F4B2F62A6E6D8E39D92
Round 9 : E43FE3BF0EC1FCF421A35A12F940C780
Round 10 : DBF92E26D538D2D2F49B88C00DDB4F40

```

SAES Keygen:

CODE:

```
import java.util.Scanner;

class KeyGeneration{
    static Scanner sc;
    static int i, j, k;
    public static void main(String[] args) {
        sc = new Scanner(System.in);
        //      S-BOX
        int sBox[][] = {{9,4,10,11}, {13,1,8,5}, {6,2,0,3}, {12,14,15,7}};
        //      Round Constant
        int[] rCon1 = {8, 0};
        int[] rCon2 = {3, 0};

        int[] k0 = key0();
        System.out.println("Second key(k1) :");
        int[] k1 = key1(k0, sBox, rCon1);
        System.out.println("Third key(k2) :");
        key2(k1, sBox, rCon2);
    }

    // Generating First key
```

```

static int[] key0() {
    System.out.println("Enter the original key");
    String[] key = new String[4];
    int[] k = new int[4];
    for(i = 0; i < 4; i++)
        key[i] = sc.next();
    System.out.println();
    System.out.println("First key(k0) :");
    for(i = 0; i < 4; i++) {
        k[i] = Integer.parseInt(key[i], 16);
        System.out.print(key[i]);
    }
    System.out.println();
    return k;
}

```

// Generating Second key

```

static int[] key1(int[] k0, int[][] sBox, int[] rCon1) {
    int[] w = new int[2];
    int[] w0 = new int[2];
    int[] w1 = new int[2];
    int[] w2 = new int[2];
    int[] w3 = new int[2];
    String[] key0 = new String[4];

    for(i = 0; i < 2; i++)
        w0[i] = k0[i];
    for(i = 2, j = 0; i < 4; i++, j++) {

```

```

        w1[j] = k0[i];
        w[j] = k0[i];
    }

    //    Rotation Nibble
    int[] rot = rotNib(w);
    //    Substitution Nibble
    int[] sub = subNib(rot, sBox);

    for(i = 0, j = 2; i < 2; i++, j++) {
        w2[i] = w0[i] ^ rCon1[i] ^ sub[i];
        w3[i] = w2[i] ^ w1[i];
        k0[i] = w2[i];
        k0[j] = w3[i];
    }

    for(i = 0; i < 4; i++) {
        key0[i] = Integer.toHexString(k0[i]);
        System.out.print(key0[i]);
    }

    System.out.println();

    return k0;
}

//Generating Third key
static void key2(int[] k1, int[][] sBox, int[] rCon2) {
    int[] w = new int[2];
    int[] w2 = new int[2];
    int[] w3 = new int[2];

```

```

int[] w4 = new int[2];
int[] w5 = new int[2];
String[] key1 = new String[4];

for(i = 0; i < 2; i++)
    w2[i] = k1[i];
for(i = 2, j = 0; i < 4; i++, j++) {
    w3[j] = k1[i];
    w[j] = k1[i];
}

//      Rotation Nibble
int[] rot = rotNib(w);
//      Substitution Nibble
int[] sub = subNib(rot, sBox);

for(i = 0, j = 2; i < 2; i++, j++) {
    w4[i] = w2[i] ^ rCon2[i] ^ sub[i];
    w5[i] = w4[i] ^ w3[i];
    k1[i] = w4[i];
    k1[j] = w5[i];
}
for(i = 0; i < 4; i++) {
    key1[i] = Integer.toHexString(k1[i]);
    System.out.print(key1[i]);
}
System.out.println();

```

```

}

//    Rotation
static int[] rotNib(int[] w) {
    int hold = w[0];
    w[0] = w[1];
    w[1] = hold;

    return w;
}

//Substitution
static int[] subNib(int[] rot, int[][] sBox) {
    int value, count;
    for(i = 0; i < 2; i++) {
        count = 0;
        value = rot[i];
        for(j = 0; j < 4; j++) {
            for(k = 0; k < 4; k++) {
                if(value == count) {
                    rot[i] = sBox[j][k];
                }
                count++;
            }
        }
    }
    return rot;
}

```

}

```
saddy@SkSaddy:~/Documents/DOCs/FALL_SEMESTER_2019-2020/CSE1007/Assignment/S-AES$ java KeyGeneration
Enter the original key
4
a
f
5

First key(k0) :
4af5
Second key(k1) :
dd28
Third key(k2) :
87af
```


SAES Encryption:

Code:

```
import java.util.*;

class S_AES_encryption {
    static int i, j, i1, j1, x, k;
    static Scanner sc;

    public static void main(String[] args) {
        sc = new Scanner(System.in);
        // S-BOX
        int sBox[][] = {{9,4,10,11}, {13,1,8,5}, {6,2,0,3}, {12,14,15,7}};
        // Temporary matrix
```

```
int temp[][] = {{1, 4}, {4, 1}};  
// Assuming Irreducible polynomial: " x^4+x+1 "
```

```
int[][] pText=plainText();  
int[][] key0 = firstKey();  
int[][] key1 = secondKey();  
int[][] key2 = thirdKey();
```

```
System.out.println("\nAdd Round Key0");  
int[][] rKey0 = addRoundKey1(pText, key0);
```

```
System.out.println("\nNibble Substitution");  
int[][] nibSub1 = nibbleSubn1(rKey0, sBox);
```

```
System.out.println("\nShift Row");  
int[][] shftR1 = shiftRow1(nibSub1);
```

```
System.out.println("\nMix Column");  
int[][] mxCol = mixColumn(shftR1, temp);
```

```
System.out.println("\nAdd Round Key1");  
int[][] rKey1 = addRoundKey2(mxCol, key1);
```

```
System.out.println("\nNibble Substitution");  
int[][] nibSub2 = nibbleSubn2(rKey1, sBox);
```

```
System.out.println("\nShift Row");
```

```

        int[][] shftR2 = shiftRow2(nibSub2);

        System.out.println("\nAdd Round Key2");
        int[][] rKey2 = addRoundKey3(shftR2, key2);

        System.out.println("\nThe Cipher Text is:");
        cipherText(rKey2);
    }

    // plain text
    static int[][] plainText() {
        System.out.println("\nEnter the plain Text");
        int p_text[] = new int[4];
        for(i = 0; i < 4; i++)
            p_text[i] = Integer.parseInt(sc.next(), 16);

        //      in Matrix form
        int plain[][] = new int[2][2];
        x = 0;
        for(i = 0; i < 2; i++){
            for(j = 0; j < 2; j++) {
                plain[i][j] = p_text[x];
                System.out.print(plain[i][j] + " ");
                x+=2;
            }
            x=1;
            System.out.println();
        }
    }

```

```

    }
    return plain;
}

//k0

static int[][] firstKey() {
    System.out.println("Enter k0");
    int[] k0 = new int[4];
    for(i = 0; i < 4; i++)
        k0[i] = Integer.parseInt(sc.next(), 16);

    //    in Matrix form
    int key0[][] = new int[2][2];
    x = 0;
    for(i = 0; i < 2; i++){
        for(j = 0; j < 2; j++) {
            key0[i][j] = k0[x];
            System.out.print(key0[i][j] + " ");
            x+=2;
        }
        x=1;
        System.out.println();
    }
    return key0;
}

// k1

static int[][] secondKey() {
    System.out.println("Enter k1");

```

```

int[] k1 = new int[4];
for(i = 0; i < 4; i++)
    k1[i] = Integer.parseInt(sc.next(), 16);

//      in Matrix form
int key1[][] = new int[2][2];
x = 0;
for(i = 0; i < 2; i++){
    for(j = 0; j < 2; j++) {
        key1[i][j] = k1[x];
        System.out.print(key1[i][j] + " ");
        x+=2;
    }
    x=1;
    System.out.println();
}
return key1;
}

// k2
static int[][] thirdKey() {
    System.out.println("Enter k2");
    int[] k2 = new int[4];
    for(i = 0; i < 4; i++)
        k2[i] = Integer.parseInt(sc.next(), 16);

    //      in Matrix form
    int key2[][] = new int[2][2];

```

```

        x = 0;
        for(i = 0; i < 2; i++){
            for(j = 0; j < 2; j++) {
                key2[i][j] = k2[x];
                System.out.print(key2[i][j] + " ");
                x+=2;
            }
            x=1;
            System.out.println();
        }
        return key2;
    }

//    Add round key0
static int[][] addRoundKey1(int[][] pText, int[][] key0) {
    for(i = 0; i < 2; i++) {
        for(j = 0; j < 2; j++) {
            pText[i][j] = pText[i][j] ^ key0[i][j];           // XOR
            System.out.print(pText[i][j] + " ");
        }
        System.out.println();
    }
    return pText;
}

//    1st Nibble Substitution
static int[][] nibbleSubn1(int[][] rKey0, int[][] sBox) {
    int value, count;
    for(i = 0; i < 2; i++) {

```

```

        for(j = 0; j < 2; j++) {
            count = 0;
            value = rKey0[i][j];
            for(i1 = 0; i1 < 4; i1++) {
                for(j1 = 0; j1 < 4; j1++) {
                    if(value == count) {
                        rKey0[i][j] = sBox[i1][j1];
                    }
                    count++;
                }
            }
            System.out.print(rKey0[i][j] + " ");
        }
        System.out.println();
    }
    return rKey0;
}

// 1st Shift Row
static int[][] shiftRow1(int[][] nibSub1) {
    int change = nibSub1[1][0];
    nibSub1[1][0] = nibSub1[1][1];
    nibSub1[1][1] = change;

    for(i = 0; i < 2; i++) {
        for(j = 0; j < 2; j++) {
            System.out.print(nibSub1[i][j] + " ");
        }
    }
}

```

```

        System.out.println();
    }
    return nibSub1;
}

// Mix Column
static int[][] mixColumn(int[][] shftR1, int[][] temp) {
    String s[][] = new String[2][2];
    String t[][] = new String[2][2];

    for(i = 0; i < 2; i++){
        for(j = 0; j < 2; j++) {
            s[i][j] = String.format("%4s",
Integer.toBinaryString(shftR1[i][j])).replace(" ", "0");
            t[i][j] = String.format("%4s",
Integer.toBinaryString(temp[i][j])).replace(" ", "0");
        }
    }

    for(i = 0; i < 2; i++) {
        for(j = 0; j < 2; j++) {
            int x = multiply(t[i][0], s[0][j]);
            int y = multiply(t[i][1], s[1][j]);
            shftR1[i][j] = x ^ y;
            System.out.print(shftR1[i][j] + " ");
        }
        System.out.println();
    }
    return shftR1;
}

```



```

}
//      Add round key1
static int[][] addRoundKey2(int[][] mxCol, int[][] key1) {
    for(i = 0; i < 2; i++) {
        for(j = 0; j < 2; j++) {
            mxCol[i][j] = mxCol[i][j] ^ key1[i][j];           // XOR
            System.out.print(mxCol[i][j] + " ");
        }
        System.out.println();
    }
    return mxCol;
}

//      2nd Nibble Substitution
static int[][] nibbleSubn2(int[][] rKey1, int[][] sBox) {
    int value, count;
    for(i = 0; i < 2; i++) {
        for(j = 0; j < 2; j++) {
            count = 0;
            value = rKey1[i][j];
            for(i1 = 0; i1 < 4; i1++){
                for(j1 = 0; j1 < 4; j1++) {
                    if(value == count) {
                        rKey1[i][j] = sBox[i1][j1];
                    }
                    count++;
                }
            }
        }
    }
}

```

```

        System.out.print(rKey1[i][j] + " ");
    }
    System.out.println();
}
return rKey1;
}
//    2nd Shift Row
static int[][] shiftRow2(int[][] nibSub2) {
    int change = nibSub2[1][0];
    nibSub2[1][0] = nibSub2[1][1];
    nibSub2[1][1] = nibSub2[1][0];

    for(i = 0; i < 2; i++) {
        for(j = 0; j < 2; j++) {
            System.out.print(nibSub2[i][j] + " ");
        }
        System.out.println();
    }
    return nibSub2;
}
//    Add round key2
static int[][] addRoundKey3(int[][] shftR2, int[][] key2) {
    for(i = 0; i < 2; i++) {
        for(j = 0; j < 2; j++) {
            shftR2[i][j] = shftR2[i][j] ^ key2[i][j];        // XOR
            System.out.print(shftR2[i][j] + " ");
        }
    }
}

```

```

        System.out.println();
    }
    return shftR2;
}

//    The Cipher Text
static void cipherText(int[][] rKey2){
    String[] cipher = new String[4];
    k = 0;
    for(i = 0; i < 2; i++) {
        for(j = 0; j < 2; j++, k+=2)
            cipher[k] = Integer.toHexString(rKey2[i][j]);
        k = 1;
    }
    for(k = 0; k < 4; k++)
        System.out.print(cipher[k]);
    System.out.println();
}

//    Left Shift for mix Column
static String leftShift(String a){
    String s="";
    for(int i = 1; i < 4; i++) {
        s = s + a.charAt(i);
    }
    s=s+0;
    return s;
}

```

```

//      Multiplication formix column
static int multiply(String t, String s) {
    String l;
    int a, add;
    String p[] = new String[4];

    p[0] = s;
    l = "";
    for(k = 1; k < 4; k++) {
        l = leftShift(p[k-1]);
        if(p[k-1].charAt(0) == '1') {
            a = Integer.parseInt(l, 2) ^ 3;
            p[k] = String.format("%4s", Integer.toBinaryString(a)).replace(" ",
"0");

        }
        else
            p[k] = l;
    }

    add = 0;
    for(k = 0; k < 4; k++) {
        if(t.charAt(k) == '1')
            add = add ^ Integer.parseInt(p[3-k], 2);
    }
    return add;
}
}

```

OUTPUT:

```
saddy@SkSaddy:~/Documents/DOCs/FALL_SEMESTER_2019-2020/CSE1007/Assignment/S-AES$ java S_AES_encryption
```

```
Enter the plain Text
```

```
d
7
2
8
13 2
7 8
Enter k0
```

```
4
a
f
5
4 15
10 5
Enter k1
```

```
d
d
2
8
13 2
13 8
Enter k2
```

```
8
7
a
f
8 10
7 15
```

```
Add Round Key0
```

```
9 13
13 13
```

```
Add Round Key0
```

```
9 13
13 13
```

```
Nibble Substitution
```

```
2 14
14 14
```

```
Shift Row
```

```
2 14
14 14
```

```
Mix Column
```

```
15 3
6 3
```

```
Add Round Key1
```

```
2 1
11 11
```

```
Nibble Substitution
```

```
10 4
3 3
```

```
Shift Row
```

```
10 4
3 3
```

```
Add Round Key2
```

```
2 14
4 12
```

```
The Cipher Text is :
```

```
24ec
```

```
saddy@SkSaddy:~/Documents/DOCs/FALL_SEMESTER_2019-2020/CSE1007/Assignment/S-AES$
```

SAES Decryption: Code:

```
import java.util.*;

class S_AES_decryption {
    static int i, j, i1, j1, x, k;
    static Scanner sc;

    public static void main(String[] args) {
        sc = new Scanner(System.in);
        // S-BOX
        int sBoxInv[][] = {{10,5,9,11}, {1,7,8,15}, {6,0,2,3}, {12,4,13,14}};
        // Temporary matrix
        int temp[][] = {{9, 2}, {2, 9}};
        // Assuming Irreducible polynomial: " x^4+x+1 "

        int[][] cText = cipherText();
        int[][] key0 = firstKey();
        int[][] key1 = secondKey();
        int[][] key2 = thirdKey();

        System.out.println("\nAdd Round Key2");
        int[][] rKey2 = addRoundKey3(cText, key2);

        System.out.println("\nInverse ShiftRow");
        int[][] invShftR2 = invShiftRow2(rKey2);

        System.out.println("\nNibble Substitution");
        int[][] invNibSub2 = invNibbleSubn2(invShftR2, sBoxInv);

        System.out.println("\nAdd Round Key1");
        int[][] rKey1 = addRoundKey2(invNibSub2, key1);

        System.out.println("\ninverse Mix Column");
        int[][] invMxCol = invMixColumn(rKey1, temp);

        System.out.println("\nInverse ShiftRow");
        int[][] invShftR1 = invShiftRow1(invMxCol);

        System.out.println("\nNibble Substitution");
        int[][] invNibSub1 = invNibbleSubn1(invShftR1, sBoxInv);

        System.out.println("\nAdd Round Key0");
        int[][] rKey0 = addRoundKey1(invNibSub1, key0);

        System.out.println("\nThe Plain Text is :");
        plainText(rKey2);
    }
}
```

```

// cipher text
static int[][] cipherText() {
    System.out.println("\nEnter the cipher Text");
    int c_text[] = new int[4];
    for(i = 0; i < 4; i++)
        c_text[i] = Integer.parseInt(sc.next(), 16);

    //      in Matrix form
    int cipher[][] = new int[2][2];
    x = 0;
    for(i = 0; i < 2; i++){
        for(j = 0; j < 2; j++) {
            cipher[i][j] = c_text[x];
            System.out.print(cipher[i][j] + " ");
            x+=2;
        }
        x=1;
        System.out.println();
    }
    return cipher;
}

// k0
static int[][] firstKey() {
    System.out.println("Enter k0");
    int[] k0 = new int[4];
    for(i = 0; i < 4; i++)
        k0[i] = Integer.parseInt(sc.next(), 16);

    //      in Matrix form
    int key0[][] = new int[2][2];
    x = 0;
    for(i = 0; i < 2; i++){
        for(j = 0; j < 2; j++) {
            key0[i][j] = k0[x];
            System.out.print(key0[i][j] + " ");
            x+=2;
        }
        x=1;
        System.out.println();
    }
    return key0;
}

// k1
static int[][] secondKey() {
    System.out.println("Enter k1");
    int[] k1 = new int[4];
    for(i = 0; i < 4; i++)
        k1[i] = Integer.parseInt(sc.next(), 16);

    //      in Matrix form
    int key1[][] = new int[2][2];

```

```

        x = 0;
        for(i = 0; i < 2; i++){
            for(j = 0; j < 2; j++) {
                key1[i][j] = k1[x];
                System.out.print(key1[i][j] + " ");
                x+=2;
            }
            x=1;
            System.out.println();
        }
        return key1;
    }
    // k2
    static int[][] thirdKey() {
        System.out.println("Enter k2");
        int[] k2 = new int[4];
        for(i = 0; i < 4; i++)
            k2[i] = Integer.parseInt(sc.next(), 16);

        //      in Matrix form
        int key2[][] = new int[2][2];
        x = 0;
        for(i = 0; i < 2; i++){
            for(j = 0; j < 2; j++) {
                key2[i][j] = k2[x];
                System.out.print(key2[i][j] + " ");
                x+=2;
            }
            x=1;
            System.out.println();
        }
        return key2;
    }
    //      Add round key0
    static int[][] addRoundKey1(int[][] cText, int[][] key0) {
        cText[0][0] = 13;
        cText[0][1] = 2;
        cText[1][0] = 7;
        cText[1][1] = 8;
        for(i = 0; i < 2; i++) {
            for(j = 0; j < 2; j++) {
                System.out.print(cText[i][j] + " ");
            }
            System.out.println();
        }
        return cText;
    }
    //      1st Nibble Substitution
    static int[][] invNibbleSubn1(int[][] rKey0, int[][] sBoxInv) {
        int value, count;
        for(i = 0; i < 2; i++) {

```



```

        for(j = 0; j < 2; j++) {
            count = 0;
            value = rKey0[i][j];
            for(i1 = 0; i1 < 4; i1++) {
                for(j1 = 0; j1 < 4; j1++) {
                    if(value == count) {
                        rKey0[i][j] = sBoxInv[i1][j1];
                    }
                    count++;
                }
            }
            System.out.print(rKey0[i][j] + " ");
        }
        System.out.println();
    }
    return rKey0;
}

// 1st Inverse ShiftRow
static int[][] invShiftRow1(int[][] invNibSub1) {
    int change = invNibSub1[1][0];
    invNibSub1[1][0] = invNibSub1[1][1];
    invNibSub1[1][1] = change;

    for(i = 0; i < 2; i++) {
        for(j = 0; j < 2; j++) {
            System.out.print(invNibSub1[i][j] + " ");
        }
        System.out.println();
    }
    return invNibSub1;
}

// invMix Column
static int[][] invMixColumn(int[][] invShftR1, int[][] temp) {
    String s[][] = new String[2][2];
    String t[][] = new String[2][2];

    for(i = 0; i < 2; i++){
        for(j = 0; j < 2; j++) {
            s[i][j] = String.format("%4s", Integer.toBinaryString(invShftR1[i][j])).replace(" ",
"0");

            t[i][j] = String.format("%4s", Integer.toBinaryString(temp[i][j])).replace(" ", "0");
        }
    }

    for(i = 0; i < 2; i++) {
        for(j = 0; j < 2; j++) {
            int x = multiply(t[i][0], s[0][j]);
            int y = multiply(t[i][1], s[1][j]);
            invShftR1[i][j] = x ^ y;
            System.out.print(invShftR1[i][j] + " ");
        }
    }
}

```

```

        System.out.println();
    }
    return invShftR1;
}
//      Add round key1
static int[][] addRoundKey2(int[][] invMxCol, int[][] key1) {
    for(i = 0; i < 2; i++) {
        for(j = 0; j < 2; j++) {
            invMxCol[i][j] = invMxCol[i][j] ^ key1[i][j];           // XOR
            System.out.print(invMxCol[i][j] + " ");
        }
        System.out.println();
    }
    return invMxCol;
}
//      2nd Nibble Substitution
static int[][] invNibbleSubn2(int[][] rKey1, int[][] sBoxInv) {
    int value, count;
    for(i = 0; i < 2; i++) {
        for(j = 0; j < 2; j++) {
            count = 0;
            value = rKey1[i][j];
            for(i1 = 0; i1 < 4; i1++) {
                for(j1 = 0; j1 < 4; j1++) {
                    if(value == count) {
                        rKey1[i][j] = sBoxInv[i1][j1];
                    }
                    count++;
                }
            }
            System.out.print(rKey1[i][j] + " ");
        }
        System.out.println();
    }
    return rKey1;
}
//      2nd Inverse ShiftRow
static int[][] invShiftRow2(int[][] rKey2) {
    int change = rKey2[1][0];
    rKey2[1][0] = rKey2[1][1];
    rKey2[1][1] = rKey2[1][0];

    for(i = 0; i < 2; i++) {
        for(j = 0; j < 2; j++) {
            System.out.print(rKey2[i][j] + " ");
        }
        System.out.println();
    }
    return rKey2;
}
//      Add round key2

```

```

static int[][] addRoundKey3(int[][] invShftR2, int[][] key2) {
    for(i = 0; i < 2; i++) {
        for(j = 0; j < 2; j++) {
            invShftR2[i][j] = invShftR2[i][j] ^ key2[i][j];           // XOR
            System.out.print(invShftR2[i][j] + " ");
        }
        System.out.println();
    }
    return invShftR2;
}

//      The Plain Text
static void plainText(int[][] rKey2) {
    String[] plain = new String[4];
    k = 0;
    for(i = 0; i < 2; i++) {
        for(j = 0; j < 2; j++, k+=2)
            plain[k] = Integer.toHexString(rKey2[i][j]);
        k = 1;
    }
    for(k = 0; k < 4; k++)
        System.out.print(plain[k]);
    System.out.println();
}

//      Left Shift for invMix Column
static String leftShift(String a) {
    String s="";
    for(int i = 1; i < 4; i++) {
        s = s + a.charAt(i);
    }
    s = s + 0;
    return s;
}

//      Multiplication for invMix column
static int multiply(String t, String s) {
    String l;
    int a, add;
    String p[] = new String[4];

    p[0] = s;
    l = "";
    for(k = 1; k < 4; k++) {
        l = leftShift(p[k-1]);
        if(p[k-1].charAt(0) == '1') {
            a = Integer.parseInt(l, 2) ^ 3;
            p[k] = String.format("%4s", Integer.toBinaryString(a)).replace(" ", "0");
        }
        else
            p[k] = l;
    }
    add = 0;
}

```

```

        for(k = 0; k < 4; k++) {
            if(t.charAt(k) == '1')
                add = add ^ Integer.parseInt(p[3-k], 2);
        }
        return add;
    }
}

```

OUTPUT:

```

saddy@SkSaddy:~/Documents/DOCs/FALL_SEMESTER_2019-2020/CSE1007/Assignment/S-AES$ java S_AES_decryption
Enter the cipher Text
2
4
e
c
2 14
4 12
Enter k0
8
7
a
f
8 10
7 15
Enter k1
d
d
2
8
13 2
13 8
Enter k2
4
a
f
5
4 15
10 5

Add Round Key2
6 1
14 9

```

Add Round Key2

6 1
14 9

Inverse ShiftRow

6 1
9 9

Nibble Substitution

8 5
0 0

Add Round Key1

5 7
13 8

inverse Mix Column

2 9
5 10

Inverse ShiftRow

2 9
10 5

Nibble Substitution

9 0
2 7

Add Round Key0

13 2
7 8

The Plain Text is :
d728

RSA KEY GENERATION

Code:

```
package rsa;
import java.util.*;
public class RSA {
    RSA()
    {
        this.keygen();
    }
    static Scanner br = new Scanner(System.in);
    public static int[] public_key = new int[2];
    public static int private_key;
    public void keygen()
    {
        this.main(new String[1]);
    }
    public void main(String[] args)
    {
        System.out.println(""Enter the value of p"");
        int p = br.nextInt();
        System.out.println(""Enter the value of q"");
        int q = br.nextInt();
        primechecker(p,q);
        int n = p*q;
        int phi_n = (p-1)*(q-1);
        possible_E(phi_n);
        System.out.println(""Enter the value of e"");
        int e = br.nextInt();
        int d = GCD(phi_n,e,true);
        public_key[0] = e;
        public_key[1] = n;
        private_key = d;
        System.out.println(""N: " + n);
        System.out.println(""Public Key: " + e);
        System.out.println(""Private Key: " + d);
    }
    public void primechecker(int p , int q)
    {
        for(int i = 2 ; i <= p/2 ; i++)
            if(p%i == 0)
                System.exit(0);
        for(int i = 2 ; i <= q/2 ; i++)
            if(q%i == 0)
                System.exit(0);
    }
    public void possible_E(int n)
    {
        for(int i = 2 ; i <= n ; i++)
```

```

if(GCD(n,i,false) == 1)
System.out.print(i + " ");
System.out.println();
}
public int GCD(int a , int b , boolean inv)
{
int s1 = 1 ;
int s2 = 0 ;
int t1 = 0 ;
int t2 = 1 ;
int q = a/b;
int ad = a;
int bd = b ;
int r = a%b ;
int t = 0 ;
while(r > 0)
{
a = b;
b = r;
t = s1 ;
s1 = s2;
s2 = t - q*s2;
t = t1 ;
t1 = t2;
t2 = t - q*t2;
r = a %b;
q = a /b ;
}
if(inv && t2 > 0)
return t2;
else if(inv && t2 < 0)
return t2+ad;
else
return b;
}
}

```

RSA Encryption

Code:

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package rsa;
/**
 *
 * @author Ak
 */
public class RSA_Encryption
{
    public static void main(String args[])
    {
        RSA key = new RSA();
        int c = Fast_Expo(5,key);
        System.out.print(""Cipher Text: " + c);
    }
    public static int Fast_Expo(int P , RSA obj)
    {
        int x = obj.public_key[0];
        int n = obj.public_key[1];
        System.out.println(n + " " + x);
        int[] binx = dectobin(x);
        int y = 1;
        for(int i = 0 ; i < binx.length ; i++)
        {
            if(binx[binx.length-i-1] == 1)
            y = mod(y * P , n);
            P = mod(P*P , n);
        }
        return y;
    }
    public static int mod(int a , int n)
    {
        return a%n;
    }
    public static int[] dectobin(int n)
    {
        String s = Integer.toBinaryString(n);
        System.out.println(s);
        int[] bin = new int[s.length()];
        for(int i = 0 ; i < s.length() ; i++)
        {
            bin[i] = s.charAt(i) - 48;
        }
    }
}
```



```
return bin;
}
}
```

OUTPUT:

```
Enter the value of p
7
Enter the value of q
11
7 11 13 17 19 23 29 31 37 41 43 47 49 53 59
Enter the value of e
13
N: 77
Public Key: 13
Private Key: 37
77 13
1101
Cipher Text: 26BUILD SUCCESSFUL (total time: 7 seconds)
|
```

RSA Decryption

Code:

```
package rsa;
public class RSA_Decryption
{
    public static void main(String agrs[])
    {
        RSA key = new RSA();
        int c = Fast_Expo(26,key);
        System.out.println("" PlainText: " + c);
    }
    public static int Fast_Expo(int P , RSA obj)
    {
        int x = obj.private_key;
        int n = obj.public_key[1];
        int[] binx = dectobin(x);
        int y = 1;
        for(int i = 0 ; i < binx.length ; i++)
        {
            if(binx[binx.length-i-1] == 1)
                y = mod(y * P , n);
            P = mod(P*P , n);
        }
        return y;
    }
    public static int mod(int a , int n)
    {
        return a%n;
    }
    public static int[] dectobin(int n)
    {
        String s = Integer.toBinaryString(n);

        int[] bin = new int[s.length()];
        for(int i = 0 ; i < s.length() ; i++)
        {
            bin[i] = s.charAt(i) - 48;
        }
        return bin;
    }
}
```

OUTPUT:

```
run:
Enter the value of p
7
Enter the value of q
11
7 11 13 17 19 23 29 31 37 41 43 47 49 53 59
Enter the value of e
13
N: 77
Public Key: 13
Private Key: 37
PlainText: 5
BUILD SUCCESSFUL (total time: 4 seconds)
|
```

MINI PROJECT ENCRYPTION CODE:

```
public class FiringSquad
{
    public static void main(String agrs[])
    {
        Minigun fire = new Minigun();
        fire.activatemain();
    }
}

//ABCDEFGHJKLMNOPQRSTUVWXYZ

public class Minigun
{
    public void activatemain()
    {
        //LEVEL ONE ENCRYPTION
        MONOADDITIVE level1 = new MONOADDITIVE();
        String level1Cipher = level1.cipher;

        //LEVEL TWO ECRYPTION
        int[] levelTwo = new int[level1Cipher.length()];
        System.out.print("Level Two Encryption: ");
        for(int i = 0 ; i < level1Cipher.length();i++)
        {
            SDES level2 = new SDES();
            levelTwo[i] = level2.main(level1Cipher.charAt(i));
        }
        for(int x:levelTwo)
            System.out.print(x+" ");
        System.out.println();
        display(levelTwo);

        //LEVEL THREE ENCRYPTION
        RSA key = new RSA();
        RSA_Encryption level3 = new RSA_Encryption(key);
        int[] levelThree = new int[levelTwo.length];

        for(int i = 0 ; i < level1Cipher.length();i++)
        {
            levelThree[i] = level3.main(levelTwo[i]);
        }

        int[] remove = new int[levelThree.length];
```

```

int i = 0 ;
for(int x:levelThree)
{
    System.out.print(x+" ");
    remove[i] = x/26;
    i++;
}
/*
System.out.println("\nPrinting removal");
for(int x : remove)
    System.out.print(x+" ");
*/
System.out.println();
display(levelThree);
}

public void display(int[] a)
{
    for(int x :a)
    {
        System.out.print((char)((x%26)+65));
    }
    System.out.println();
}
}

```

```

//47 77 97 16 230 60 331 209 173 208 196 314 268 313 292 132 323 313 383 331 3 44 92 165 373 272
//KLMUSXZMSCJJVAEYZMPNYGFQAW

```

```

import java.util.*;
public class MONOADDITIVE
{
    MONOADDITIVE()
    {
        //System.out.println("IN ADDITION");
        this.main();
    }
    Scanner br = new Scanner(System.in);
    String cipher ;
    public void main()
    {
        System.out.println("Enter the plaintext");
        String P = br.next();
        System.out.println("Enter the key for encryption: ");
        int key = br.nextInt();
        Encrypt( P /*"CRYPTOGRAPHYISFORINFORMATIONSECURITY" */ , key);
    }
    void Encrypt(String cipertext , int i)
    {
        String message = "";

```

```

for(int j = 0 ; j < cipertext.length() ; j++)
{
    char letter = cipertext.charAt(j);
    int number = converter(letter);
    number = modifier(number + i);
    letter = deconverter(number);
    message = message + letter ;
}

System.out.println("Level One Encryption : " +message);
cipher = message;
}
public int converter(char a)
{
    if(a - 65 >= 0)
        return (int)(a-65);
    else
        return (int)((a-65)+26) ;
}
public char deconverter(int a)
{
    return (char)(a + 65);
}
public int modifier(int c)
{
    if(c < 0 )
        return (c+26)%26;
    else
        return c%26;
}
}

public class SDES
{
    int[] plain = new int[8];
    int[] right = new int[4];
    int[] left = new int[4];
    int[] cipher = new int[8];
    int[][] keyfinal = {{0,0,1,0,1,1,1,1},{1,1,1,0,1,0,1,0}};
    int CipherText;
    public void permutation()
    {
        int[] p = {1,5,2,0,3,7,4,6};
        int[] a = new int[8];
        for(int i = 0 ; i< p.length ; i++)
        {
            a[i] = plain[p[i]];
        }
        plain = a;
    }
}

```

```

public void separator()
{
    for(int i = 0;i<4;i++)
    {
        left[i] = plain[i];
        right[i] = plain[i+4];
    }
}
public int[] expansion(int[]c)
{
    int[] p = {3,0,1,2,1,2,3,0};
    int[] b = new int[8];
    for(int i = 0 ; i< p.length ; i++)
    {
        b[i] = c[p[i]];
    }
    return b;
}
public void swapper()
{
    int[] a = right;
    right = left;
    left = a;
}
public int[] permutationfour(int[] a)
{
    int[] p = {1,3,2,0};
    int[] b = new int[4];
    for(int i = 0 ; i< p.length ; i++)
    {
        b[i] = a[p[i]];
    }
    return b;
}
public int[] sbox(int[] a)
{
    int[][] szero = {{1,0,3,2},{3,2,1,0},{0,2,1,3},{3,1,3,2}};
    int[][] sone = {{0,1,2,3},{2,0,1,3},{3,0,1,0},{2,1,0,3}};
    int one,two,three,four;
    one = a[0]*2 + a[3];
    two = a[1]*2 + a[2];
    three = a[4]*2 + a[7];
    four = a[5]*2 + a[6];
    int o = szero[one][two];
    int t = sone[three][four];
    int[] b = {o/2,o%2,t/2,t%2};
    return b;
}
public int[] function(int n)
{
    int[] a = expansion(right);

```

```

        for(int i = 0 ; i < 8 ; i++)
            a[i] = keyfinal[n][i]^a[i] ;
        a = sbbox(a);
        a = permutationfour(a);
        for(int i = 0 ; i < a.length ; i++)
            a[i] = a[i]^left[i];
        return a;
    }
    public void combine()
    {
        for(int i = 0 ; i < 4 ; i++)
        {
            plain[i] = left[i];
            plain[i+4] = right[i];
        }
    }
    public void ipermutation()
    {
        int[] p = {3,0,2,4,6,1,7,5};
        int[] a = new int[8];
        for(int i = 0 ; i < p.length ; i++)
        {
            a[i] = plain[p[i]];
        }
        cipher = a;
    }
    public void setplaintext(char con)
    {
        char c = con ;
        int ci = c ;
        String s = Integer.toBinaryString(ci);
        for(int i = 0 ; i < 8 ; i++)
        {
            if(i < 8 - s.length())
                plain[i] = 0;
            else
                plain[i] = (int)(s.charAt(i - (8-s.length()))) - 48;
        }
    }
    void setCipherText()
    {
        int t = 1 ;
        for(int i = 0 ; i < 8 ; i++)
        {
            CipherText = CipherText + t*cipher[7 - i];
            t=t*2;
        }
    }
    public int main(char con)
    {
        // System.out.println("IN DES");

```



```

        setplaintext(con);
        permutation();
        separator();
        left = function(0);
        swapper();
        left = function(1);
        combine();
        ipermutation();
        setCipherText();
        return CipherText;
    }
}

import java.util.*;
public class RSA {
    static Scanner br = new Scanner(System.in);
    public int[] public_key = new int[2];
    public int private_key;
    RSA()
    {
        System.out.println("Enter the value of p");
        int p = br.nextInt();
        System.out.println("Enter the value of q");
        int q = br.nextInt();
        primechecker(p,q);
        int n = p*q;
        int phi_n = (p-1)*(q-1);
        possible_E(phi_n);
        System.out.println("Enter the value of e");
        int e = br.nextInt();
        int d = GCD(phi_n,e,true);
        public_key[0] = e;
        public_key[1] = n;
        private_key = d;
        //System.out.println("N : " + n);
        //System.out.println("Public Key: " + e);
        //System.out.println("Private Key: " + d);
    }
    public void primechecker(int p , int q)
    {
        for(int i = 2 ; i < p/2 ; i++)
            if(p%i == 0)
                System.exit(0);

        for(int i = 2 ; i < q/2 ; i++)
            if(q%i == 0)
                System.exit(0);
    }
    public void possible_E(int n)
    {
        for(int i = 2 ; i < n ; i++)

```

```

        if(GCD(n,i,false) == 1)
            System.out.print(i + " ");
        System.out.println();
    }
    public int GCD(int a , int b , boolean inv)
    {
        int s1 = 1 ;
        int s2 = 0 ;
        int t1 = 0 ;
        int t2 = 1 ;
        int q = a/b;
        int ad = a;
        int r = a%b ;
        int t = 0 ;
        while(r > 0)
        {
            a = b;
            b = r;
            t = s1 ;
            s1 = s2;
            s2 = t - q*s2;
            t = t1 ;
            t1 = t2;
            t2 = t - q*t2;
            r = a %b;
            q = a /b ;
        }

        if(inv && t2 > 0)
            return t2;
        else if(inv && t2 < 0)
            return t2+ad;
        else
            return b;
    }
}

public class RSA_Encryption
{
    RSA key;
    RSA_Encryption(RSA key)
    {
        this.key = key;
    }
    public int main(int p)
    {
        int c = Fast_Expo(p,key);
        return c;
    }
}

```

```

public int Fast_Expo(int P , RSA obj)
{
    int x = obj.public_key[0];
    int n = obj.public_key[1];
    int[] binx = dectobin(x);
    int y = 1;
    for(int i = 0 ; i < binx.length ; i++)
    {

        if(binx[binx.length-i-1] == 1)
            y = mod(y * P , n);

        P = mod(P*P , n);
    }

    return y;
}
public int mod(int a , int n)
{
    return a%n;
}

public int[] dectobin(int n)
{
    String s = Integer.toBinaryString(n);
    int[] bin = new int[s.length()];
    for(int i = 0 ; i < s.length() ; i++)
    {
        bin[i] = s.charAt(i) - 48;
    }
    return bin;
}
}

```

OUTPUT:

```

Enter the plaintext
ABCDEF GHIJ KLMNOPQRST UVWXYZ
Enter the key for encryption:
1
Level One Encryption : BCDEF GHIJ KLMNOPQRST UVWXYZA
Level Two Encryption: 178 149 124 140 243 16 83 130 40 170 18 231 77 207 114 47 3 76 51 74 102 41 244 33 63 233
WTUKJQFAOOSXZZKVDYZWYPKHLZ
Enter the value of p
101
Enter the value of q
103
7 11 13 19 23 29 31 37 41 43 47 49 53 59 61 67 71 73 77 79 83 89 91 97 101 103 107 109 113 121 127 131 133 137 139 143 149 151 157 1
Enter the value of e
83
1232 2013 2534 436 5998 1583 8631 5446 4516 5410 5105 8173 6989 8138 7596 3456 8423 8150 9973 8619 102 1150 2397 4306 9698 7094
KLMUSXZMSCJJVAEYZMPNYGFQAW
|

```

MINI PROJECT DECRYPTION CODE:

```
public class RedCross
{
    public static void main(String agrs[])
    {
        Mission relief = new Mission();
        relief.activateShield();
    }
}

public class Mission
{
    public void activateShield()
    {
        int[] cipher = cipherMaker("KLMUSXZMSCJJVAEYZMPNYGFQAW" , new int[]
{47,77,97,16,230,60,331,209,173,208,196,314,268,313,292,132,323,313,383,331,3,44,92,165,373,272});
        RSA key = new RSA();
        int[] levelThreeRemoved = new int[cipher.length];
        int i = 0 ;
        for(int x: cipher)
        {
            DRSA obj = new DRSA();
            levelThreeRemoved[i] = obj.main(key,x);
            i++;
        }

        System.out.println("Level Three Removed");
        for(int x: levelThreeRemoved)
            System.out.print(x + " ");

        i = 0;
        int[] levelTwoRemoval = new int[levelThreeRemoved.length];
        for(int x : levelThreeRemoved)
        {
            DSDES obj = new DSDES();
            levelTwoRemoval[i] = obj.main(x);
            i++;
        }

        System.out.println("\nLevel Two Removed");
        for(int x: levelTwoRemoval)
            System.out.print(x + " ");

        String s = "";
```

```

        for(int x:levelTwoRemoval)
            s = s + (char)(x);

        DMonoAdditive rem = new DMonoAdditive();
        rem.main(s);
        s = rem.cipher;
        System.out.println("Plaintext: "+s);
    }

    public int[] cipherMaker(String s , int[] b)
    {
        int[] cipher = new int[b.length];
        for(int i = 0 ; i < b.length ; i++)
        {
            cipher[i] = b[i]*26 + (s.charAt(i)-65);
        }

        return cipher;
    }
}

import java.util.*;
public class RSA {
    static Scanner br = new Scanner(System.in);
    public int[] public_key = new int[2];
    public int private_key;
    RSA()
    {
        System.out.println("Enter the value of p");
        int p = br.nextInt();
        System.out.println("Enter the value of q");
        int q = br.nextInt();
        primechecker(p,q);
        int n = p*q;
        int phi_n = (p-1)*(q-1);
        possible_E(phi_n);
        System.out.println("Enter the value of e");
        int e = br.nextInt();
        int d = GCD(phi_n,e,true);
        public_key[0] = e;
        public_key[1] = n;
        private_key = d;
        //System.out.println("N : " + n);
        //System.out.println("Public Key: " + e);
        //System.out.println("Private Key: " + d);
    }
    public void primechecker(int p , int q)
    {
        for(int i = 2 ; i < p/2 ; i++)
            if(p%i == 0)
                System.exit(0);
    }
}

```

```

        for(int i = 2 ; i < q/2 ; i++)
            if(q%i == 0)
                System.exit(0);
    }
    public void possible_E(int n)
    {
        for(int i = 2 ; i < n ; i++)
            if(GCD(n,i,false) == 1)
                System.out.print(i + " ");
        System.out.println();
    }
    public int GCD(int a , int b , boolean inv)
    {
        int s1 = 1 ;
        int s2 = 0 ;
        int t1 = 0 ;
        int t2 = 1 ;
        int q = a/b;
        int ad = a;
        int r = a%b ;
        int t = 0 ;
        while(r > 0)
        {
            a = b;
            b = r;
            t = s1 ;
            s1 = s2;
            s2 = t - q*s2;
            t = t1 ;
            t1 = t2;
            t2 = t - q*t2;
            r = a %b;
            q = a /b ;
        }

        if(inv && t2 > 0)
            return t2;
        else if(inv && t2 < 0)
            return t2+ad;
        else
            return b;
    }
}

public class DRSA
{
    public int main(RSA key , int p)
    {
        int intp = p ;

```

```

        int c = Fast_Expo(intp,key);
        return c;
    }

    public int Fast_Expo(int P , RSA obj)
    {
        int x = obj.private_key;
        int n = obj.public_key[1];
        int[] binx = dectobin(x);
        int y = 1;
        for(int i = 0 ; i < binx.length ; i++)
        {
            if(binx[binx.length-i-1] == 1)
                y = mod(y * P , n);

            P = mod(P*P , n);
        }

        return y;
    }
    public int mod(int a , int n)
    {
        return a%n;
    }

    public int[] dectobin(int n)
    {
        String s = Integer.toBinaryString(n);
        int[] bin = new int[s.length()];
        for(int i = 0 ; i < s.length() ; i++)
        {
            bin[i] = s.charAt(i) - 48;
        }
        return bin;
    }
}

public class DSDES
{
    int[] plain = new int[8];
    int[] right = new int[4];
    int[] left = new int[4];
    int[] cipher = new int[8];
    int[][] keyfinal = {{1,1,1,0,1,0,1,0},{0,0,1,0,1,1,1,1}};
    int CipherText;
    public void permutation()
    {
        int[] p = {1,5,2,0,3,7,4,6};
        int[] a = new int[8];
        for(int i = 0 ; i < p.length ; i++)

```

```

        {
            a[i] = plain[p[i]];
        }
        plain = a;
    }
    public void separator()
    {
        for(int i = 0;i<4;i++)
        {
            left[i] = plain[i];
            right[i] = plain[i+4];
        }
    }
    public int[] expansion(int[]c)
    {
        int[] p = {3,0,1,2,1,2,3,0};
        int[] b = new int[8];
        for(int i = 0 ; i< p.length ; i++)
        {
            b[i] = c[p[i]];
        }
        return b;
    }
    public void swapper()
    {
        int[] a = right;
        right = left;
        left = a;
    }
    public int[] permutationfour(int[] a)
    {
        int[] p = {1,3,2,0};
        int[] b = new int[4];
        for(int i = 0 ; i< p.length ; i++)
        {
            b[i] = a[p[i]];
        }
        return b;
    }
    public int[] sbox(int[] a)
    {
        int[][] szero = {{1,0,3,2},{3,2,1,0},{0,2,1,3},{3,1,3,2}};
        int[][] sone = {{0,1,2,3},{2,0,1,3},{3,0,1,0},{2,1,0,3}};
        int one,two,three,four;
        one = a[0]*2 + a[3];
        two = a[1]*2 + a[2];
        three = a[4]*2 + a[7];
        four = a[5]*2 + a[6];
        int o = szero[one][two];
        int t = sone[three][four];
        int[] b = {o/2,o%2,t/2,t%2};
    }

```



```

        return b;
    }
    public int[] function(int n)
    {
        int[] a = expansion(right);
        for(int i = 0 ; i < 8 ; i++)
            a[i] = keyfinal[n][i]^a[i] ;
        a = sbbox(a);
        a = permutationfour(a);
        for(int i = 0 ; i < a.length ; i++)
            a[i] = a[i]^left[i];
        return a;
    }
    public void combine()
    {
        for(int i = 0 ; i < 4 ; i++)
        {
            plain[i] = left[i];
            plain[i+4] = right[i];
        }
    }
    public void ipermutation()
    {
        int[] p = {3,0,2,4,6,1,7,5};
        int[] a = new int[8];
        for(int i = 0 ; i < p.length ; i++)
        {
            a[i] = plain[p[i]];
        }
        cipher = a;
    }
    public void setplaintext(int con)
    {
        int ci = con ;
        String s = Integer.toBinaryString(ci);
        for(int i = 0 ; i < 8 ; i++)
        {
            if(i < 8 - s.length())
                plain[i] = 0;
            else
                plain[i] = (int)(s.charAt(i - (8-s.length()))) - 48);
        }
    }
    void setCipherText()
    {
        int t = 1 ;
        for(int i = 0 ; i < 8 ; i++)
        {
            CipherText = CipherText + t*cipher[7 - i];
            t=t*2;
        }
    }

```

```

    }
    public char main(int c)
    {
        setplaintext(c);
        permutation();
        separator();
        left = function(0);
        swapper();
        left = function(1);
        combine();
        ipermutation();
        setCipherText();
        return (char)CipherText;
    }
}

import java.util.Scanner;
public class DMonoAdditive
{
    Scanner br = new Scanner(System.in);
    String cipher ;
    public void main(String P)
    {
        System.out.println("\nEnter the key for decryption: ");
        int key = br.nextInt();
        Decrypt( P /*"CRYPTOGRAPHYISFORINFORMATIONSECURITY" */ , key);
    }
    void Decrypt(String cipertext , int i)
    {
        String message = "";
        for(int j = 0 ; j < cipertext.length() ; j++)
        {
            char letter = cipertext.charAt(j);
            int number = converter(letter);
            number = modifier(number - i);
            letter = deconverter(number);
            message = message + letter ;
        }

        //System.out.println("Plaintext: "+message);
        cipher = message;
    }
    public int converter(char a)
    {
        if(a - 65 >= 0)
            return (int)(a-65);
        else
            return (int)((a-65)+26) ;
    }
    public char deconverter(int a)

```

```

    {
        return (char)(a + 65);
    }
    public int modifier(int c)
    {
        if(c < 0 )
            return (c+26)%26;
        else
            return c%26;
    }
}

```

OUTPUT:

```

Enter the value of p
101
Enter the value of q
103
7 11 13 19 23 29 31 37 41 43 47 49 53 59 61 67 71 73 77 79 83 89 91 97 101 103 107 109 113 121 127 131 133
Enter the value of e
83
Level Three Removed
178 149 124 140 243 16 83 130 40 170 18 231 77 207 114 47 3 76 51 74 102 41 244 33 63 233
Level Two Removed
66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 65
Enter the key for decryption:
1
Plaintext: ABCDEFGHIJKLMNOPQRSTUVWXYZ

```