Shankar Narayanan S
312217104146

Date: 7-9-2020

Exercise 2

Hill Cipher :

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

class HillCypher{
    public String ptext;
    public String key;
    public int[][] ptmat,cmat, keymat;
    public int det;
    public int[][] adj;

    public HillCypher(String p, String k){
            this.ptext = p;
            this.key = k;
            int n=p.length();
            ptmat = new int[n][1];
            cmat = new int[n][1];
            keymat = new int[n][n];
            adj = new int[n][n];
    }

    public void fillMatrix(){
            int n=ptext.length();
            int i,j;

            for(i=0;i<n;i++){
                    ptmat[i][0] = (ptext.charAt(i)-'a');
            }

            int k=0;
            for(i=0;i<n;i++){
                    for(j=0;j<n;j++){
                            keymat[i][j] = key.charAt(k)-'a';
                            k++;
                    }
```

```
        }
}

public void encrypt(){
        int i,j;
        int n = ptext.length();

        for(i=0;i<n;i++){
                cmat[i][0]=0;
        }

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

                        cmat[i][0]+= keymat[i][j] * ptmat[j][0];
                }
                cmat[i][0] = (cmat[i][0]%26+26)%26;
        }
}

public void displayMatrix(){
        int i,j;
        int n=ptext.length();

        System.out.println("\n\nThe message vector : ");
        for(i=0;i<n;i++){
                System.out.println(ptmat[i][0]);
        }

        System.out.println("\n\nThe key matrix : ");
        for(i=0;i<n;i++){
                for(j=0;j<n;j++){
                        System.out.print(keymat[i][j]+" ");
                }
                System.out.println("\n");
        }

        StringBuffer cipher = new StringBuffer();
        System.out.println("\n\nThe Cypher matrix : ");
        for(i=0;i<n;i++){
                char ch = (char)(cmat[i][0]+97);
                cipher.append(ch);
                System.out.println(cmat[i][0]);
```

```
        }

        System.out.println("The cipher text : "+cipher+"\n");

}

//calculating determinant of keymat
public void calcDeterminant(){
        int n=ptext.length();
        int a,b,c;
        a = keymat[1][1]*keymat[2][2] - keymat[1][2]*keymat[2][1];
        b = keymat[1][0]*keymat[2][2] - keymat[2][0]*keymat[1][2];
        c = keymat[1][0]*keymat[2][1] - keymat[1][1]*keymat[2][0];

        //System.out.println("a = "+a+" b = "+b+" c = "+c+"\n");

        det = keymat[0][0]*a - keymat[0][1]*b + keymat[0][2]*c;
}

public void calcInverseKey(){
        int i,j,x,y;
        int a,b,c,d,e,f,g,h,k;
        int n=ptext.length();

        a = keymat[0][0];
        b = keymat[0][1];
        c = keymat[0][2];
        d = keymat[1][0];
        e = keymat[1][1];
        f = keymat[1][2];
        g = keymat[2][0];
        h = keymat[2][1];
        k = keymat[2][2];

        adj[0][0] = e*k - f*h;
        adj[0][1] = -(b*k - c*h);
        adj[0][2] = b*f - c*e;
        adj[1][0] = -(d*k - f*g);
        adj[1][1] = a*k - c*g;
        adj[1][2] = -(a*f - c*d);
        adj[2][0] = d*h - e*g;
        adj[2][1] = -(a*h - b*g);
        adj[2][2] = a*e - b*d;
```

```
        int inv = calculateInverseDet();

        if(inv == -1){
                System.out.println("Uh Oh! the key deosn't have an inverse!\n");
                return ;
        } else {
                System.out.println("The inverse of determinant : "+inv+"\n");
        }

        for(i=0;i<n;i++){
                for(j=0;j<n;j++){
                        adj[i][j]= ((inv*adj[i][j])%26 + 26)%26;
                }
        }

        //displaying the inverse

        //System.out.println("Displaying determinant value : "+det+"\n");

        System.out.println("Displaying the inverse key:");

        for(i=0;i<n;i++){
                for(j=0;j<n;j++){
                        System.out.print(adj[i][j]+" ");
                }
                System.out.print("\n");
        }
}

public int calculateInverseDet(){
        int i;

        for(i=1;i<26;i++){
                int val = i*det;
                int mul = (val%26+26)%26;
                if(mul == 1) return i;
        }

        return -1;
}
```

```
    public void decrypt(){
        int i,j;
        int n = ptext.length();
        StringBuffer decr = new StringBuffer();

        int sum;

        System.out.println("decrypted message matrix: ");
        for(i=0;i<n;i++){
            sum = 0;
            for(j=0;j<n;j++){
                sum+= adj[i][j]*cmat[j][0];
            }
            sum%= 26;
            char ch = (char)(97+sum);
            decr.append(ch);
            System.out.println(sum+"\n");
        }

        System.out.println("decrypted message : "+decr);
    }
}

public class HCDriver{
    public static void main(String[] args){

        Scanner in = new Scanner(System.in);
        String ptext, key;

        System.out.println("HILL CIPHER (for 3x3 keys only)\n\nEnter the message : ");
        ptext = in.nextLine();

        System.out.println("Enter the key : ");
        key= in.nextLine();

        HillCypher hc = new HillCypher(ptext, key);

        hc.fillMatrix();

        hc.encrypt();

        hc.displayMatrix();
```

```
            hc.calcDeterminant();

            hc.calcInverseKey();

            hc.decrypt();
    }
}
```

OUTPUT :

HILL CIPHER (for 3x3 keys only)

Enter the message :
act
Enter the key :
gybnqkurp

The message vector :
0
2
19

The key matrix :
6 24 1

13 16 10

20 17 15

The Cypher matrix :
15
14
7
The cipher text : poh

The inverse of determinant : 25

Displaying the inverse key:

8 5 10
21 8 21
21 12 8
decrypted message matrix:
0

2

19

decrypted message : act

Vigenere Cipher :

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

class VigenereCipher{

  public String ptext;
  public String key;
  public StringBuffer keyT, encr;

  public VigenereCipher(String k, String p){
        this.ptext = p;
        this.key = k;
  }

  public void generateKey(){
        keyT = new StringBuffer();

        int i,n = ptext.length();
        int m = key.length();
        for(i=0;i<n;i++){
        keyT.append(key.charAt(i%m));
        }
  }

  public void encrypt(){
```

```
        int i,n;
        n = ptext.length();

        encr = new StringBuffer();

        for(i=0;i<n;i++){
        int a = ptext.charAt(i) - 'a';
        int b = keyT.charAt(i) - 'a';

        char cur = (char)(97+(a+b)%26);
        encr.append(cur);
        }
}

public void decrypt(){
        int i,n;
        n = ptext.length();

        StringBuffer decr = new StringBuffer();

        for(i=0;i<n;i++){
        int a = encr.charAt(i) - 'a';
        int b = keyT.charAt(i) - 'a';

        char cur = (char)(97 + (a-b+26)%26);

        decr.append(cur);
        }

        //displaying decrypted message

        System.out.println("Decrypted message : "+decr);
}

public void displayTable(){
        System.out.println("\n\nDisplaying Vigenere Table : ");

        for(int i=0;i<26;i++){
        for(int j=i;j<i+26;j++){
        System.out.print((char)(97+j%26)+" ");
        }
        System.out.print("\n");
        }
```

```
        System.out.println("\n");
  }
}

public class VigDriver{

  public static void main(String[] args){
        Scanner in = new Scanner(System.in);

        String key, ptext;
        String ctext;

        System.out.println("Enter the Plain text (small characters only) : ");
        ptext = in.nextLine();

        System.out.println("Enter the key (small characters only) : ");
        key= in.nextLine();

        VigenereCipher vc = new VigenereCipher(key, ptext);

        vc.displayTable();

        vc.generateKey();
        System.out.println("Generated key : "+vc.keyT);

        vc.encrypt();
        System.out.println("Encrypted text : "+vc.encr);

        vc.decrypt();

  }
}
```

OUTPUT :

Enter the Plain text (small characters only) :
we are compromised
Enter the key (small characters only) :
kidding

Displaying Vigenere Table :

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

Generated key : kiddingkiddingki
Encrypted text : gmdumpuwxuruvyol

Decrypted message : wearecompromised