Author : Shankar Narayanan S
Reg No : 312217104146

Exercise 1

CaesarCipher.java

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

public class CaesarCipher{

   public static StringBuffer encrypt(String plaintext,int key,boolean dec){
        StringBuffer ans= new StringBuffer();

        if(dec) {
                key = -key;
        }
        int n = plaintext.length();
        for(int i=0;i<n;i++){
                char ch = plaintext.charAt(i);
                if (ch == ' ') {
                        ans.append(' ');
                        continue;
                }
                int no = ch-'a';
                char an = (char)((no+key+26)%26 + 97);

                ans.append(an);
        }

        return ans;
   }

   //function to apply brute-force cryptanalysis on the ciphertext
   public static void cryptAnalysis(StringBuffer ctext){
        String[] list ={"ab", "ad", "ah", "am","an", "as", "at", "au","be", "by", "do",
```

```
                                        "er","go", "ha", "hi", "is","it", "my", "no",
"of","on","or","re","to","we","ya","be","ba","at","re"};
        StringBuffer cur,mx;
        int cnt,mxcnt=0;
        int n = 30;
        int key=0;
        mx = new StringBuffer();

        System.out.println("\nCryptanalysis : \n\nDisplaying dictionary : \n");
        for(int i=0;i<30;i++){
                System.out.print(list[i]+" ");
        }
        System.out.println("\n");

        for (int k=1;k<26;k++){
                cnt=0;
                String text = ctext.toString();
                cur = encrypt(text,k,true);
                System.out.println(cur);

                for(int i=0;i<n;i++){
                        String str = cur.toString();
                        if(str.contains(list[i])){
                                cnt++;
                        }
                }

                if(cnt > mxcnt){
                        //System.out.println("cur max = "+cnt);
                        mxcnt = cnt;
                        mx = cur;
                        key=k;
                }
        }

        System.out.println("\n\nAfter cryptanalysis : \nKey = "+(key)+"\nDecrypted String
: "+mx);

    }
```

```
    public static void main(String[] args){

        String plaintext;
        int key;

        Scanner in = new Scanner(System.in);

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

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

        System.out.println("Encrypted message : ");
        StringBuffer encr = encrypt(plaintext,key,false);

        System.out.println(encr+"\n");


        System.out.println("Decrypted message : ");
        System.out.println(encrypt(encr.toString(),key,true)+"\n");

        // System.out.println("Enter message for cryptanalysis: ");
        // plaintext = in.nextLine();

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

        encr = encrypt(encr.toString(),0,false);

        cryptAnalysis(encr);

    }
}
```

OUTPUT :

Enter the plaintext :
i am the batman
Enter the key :
3
Encrypted message :
l dp wkh edwpdq

Decrypted message :
i am the batman

Cryptanalysis :

Displaying dictionary :

ab ad ah am an as at au be by do er go ha hi is it my no of on or re to we ya be ba at re

k co vjg dcvocp
j bn uif cbunbo
i am the batman
h zl sgd azslzm
g yk rfc zyrkyl
f xj qeb yxqjxk
e wi pda xwpiwj
d vh ocz wvohvi
c ug nby vunguh
b tf max utmftg
a se lzw tslesf
z rd kyv srkdre
y qc jxu rqjcqd
x pb iwt qpibpc
w oa hvs pohaob
v nz gur ongzna
u my ftq nmfymz
t lx esp mlexly

s kw dro lkdwkx
r jv cqn kjcvjw
q iu bpm jibuiv
p ht aol ihathu
o gs znk hgzsgt
n fr ymj gfyrfs
m eq xli fexqer

After cryptanalysis :
Key = 3
Decrypted String : i am the batman

PFDriver.java

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

class PlayfairCipher{

    public String key;
    public String ptext;
    public char[][] keyT;
    //keyn is the prepared text of the plaintext
    public StringBuffer keyn;
    public int x1,x2,y1,y2;

    public PlayfairCipher(String k,String p){
            this.key = k;
            this.ptext = p;
            keyn = new StringBuffer();
            keyT = new char[5][5];
    }

    public void generateKey(){
            int[] rem = new int[27];
            int i,j;
```

```
        for(i=0;i<26;i++){
                rem[i]=1;
        }

        i=0; j=0;
        int k=0;
        int ks = key.length();
        for(k=0;k<ks;k++){
                int ind = key.charAt(k)-'a';
                //System.out.println(key.charAt(k));
                if(rem[ind] == 1){
                        if(ind==8 || ind==9) {
                                keyT[i][j] = 'i';
                                rem[8]=0;
                                rem[9]=0;
                        }
                        else {
                                keyT[i][j] = key.charAt(k);
                        }
                        rem[ind]=0;
                        j++;

                        if(j==5){
                                i++;
                                j=0;
                        }
                }
        }

        for(k=0;k<26;k++){
                if(rem[k] != 0){
                        if(k==9) continue;
                        char ch = (char)(97+k);
                        keyT[i][j] = ch;
                        j++;

                        if(j==5){
                                i++;
                                j=0;
                        }
                }
        }
}
```

```
public void displayKeyT(){
	System.out.println("Displaying Key Table : ");

	for(int i=0;i<5;i++){
		for(int j=0;j<5;j++){
			System.out.print(keyT[i][j]+" ");
		}
		System.out.println();
	}
}

public void prepareText(){
	int i=0;

	int n = ptext.length();
	while(i < n){
		if(i == n-1){
			keyn.append(ptext.charAt(n-1));
			keyn.append('x');
			break;
		}

		char a,b;
		a = ptext.charAt(i);
		b = ptext.charAt(i+1);

		if(a==b){
			keyn.append(a);
			keyn.append('x');
			i+=1;
		} else {
			keyn.append(a);
			keyn.append(b);
			i+=2;
		}


	}
}

public void search(char a, char b){
	int i,j;
```

```
        for(i=0;i<5;i++){
                for(j=0;j<5;j++){

                        if(keyT[i][j] == a){
                                x1=i;
                                y1=j;
                        }

                        if(keyT[i][j] == b){
                                x2=i;
                                y2=j;
                        }
                }
        }
}

public StringBuffer encrypt(boolean decrypt, StringBuffer pt){
        int n,i,j;
        n = pt.length();
        StringBuffer ans = new StringBuffer();

        int p=1;
        if(decrypt) p=-1;

        for(i=0;i<n;i+=2){
                char a,b;
                a = pt.charAt(i);
                b = pt.charAt(i+1);

                search(a,b);

                if(x1 == x2){
                        ans.append(keyT[x1][(y1+p+5)%5]);
                        ans.append(keyT[x2][(y2+p+5)%5]);
                } else if(y1 == y2){
                        ans.append(keyT[(x1+p+5)%5][y1]);
                        ans.append(keyT[(x2+p+5)%5][y2]);
                } else {
                        ans.append(keyT[x1][y2]);
                        ans.append(keyT[x2][y1]);
                }
        }
```

```
        return ans;
    }
}

public class PFDriver{

    public static void main(String []args){

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

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

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

        PlayfairCipher pf = new PlayfairCipher(key, ptext);

        pf.prepareText();

        System.out.println("\n\nprepared text : "+pf.keyn);

        pf.generateKey();

        pf.displayKeyT();

        StringBuffer encr = pf.encrypt(false,pf.keyn);
        System.out.println("Encrypted message : "+encr);

        StringBuffer decr = pf.encrypt(true,encr);
        System .out.println("Decrypted message : "+decr);

    }
}
```

OUTPUT:

Enter the plaintext :
instruments
Enter the key :
monarchy

prepared text : instrumentsx
Displaying Key Table :
m o n a r
c h y b d
e f g i k
l p q s t
u v w x z
Encrypted message : gatlmzclrqxa
Decrypted message : instrumentsx