# ISC ASSIGNMENT -04

ROLL NO: U21CS052
NAME : PANCHAL GUNGUN PARESH

## PLAYFAIR CIPHER

**QUE1 .Design and implement a program to perform encryption and decryption using the Playfair Cipher with both a 5x5 and a 6x6 matrix. Consider the following inputs for the program.**

**5 x 5 matrix**

```
import java.util.*;

public class p1_playfair_5_5 {
    public static Character key_matrix[][];
    public static HashMap<Character,Cordinate>get_cordinate;
    public static HashSet<Character> hs;
    public static void init(){
        key_matrix=new Character[5][5];
        get_cordinate=new HashMap<>();
        hs=new HashSet<>();
        // System.out.println(hs);
        // print(key_matrix);

    }

    public static void print(Character arr[][]){
        for(int i=0;i<arr.length;i++){
            for(int j=0;j<arr.length;j++){
            System.out.print(arr[i][j]+"  ");
            }
            System.out.println();
```

```
        }
    }
    public static class Cordinate{
        int x,y;
        Cordinate(int x,int y){this.x=x;this.y=y;}
    }

    public static void generate_key_matrix(String key_phrase){
        /*keep a track of the characters which have already
occured. */

        int n=key_phrase.length(),i=0,j=0;
        key_phrase= key_phrase.toLowerCase();
        // System.out.println("n : "+n+" i: "+i+" j: "+j+"
key_phrase"+key_phrase);

        for(int idx=1;idx<=n;idx++){
            char ch=key_phrase.charAt(idx-1);
            // System.out.println("idx: "+idx+" ch: "+ch+" i :
"+i+" j: "+j);
            if(!hs.contains(ch)){
                /*ignore if it is a space character */
                if(ch==' '){
                    // System.out.println("----------- space
------");
                    continue;
                }
                if(ch=='i' || ch=='j'){
                    hs.add('i');hs.add('j');
                    get_cordinate.put('i', new Cordinate(i,
j));
                    // System.out.println("ch == > "+ch);
                    get_cordinate.put('j', new Cordinate(i,
j));
```

```
                }
                key_matrix[i][j]=ch;
                get_cordinate.put(ch, new Cordinate(i, j));

                hs.add(ch);
                // if(i>=5 || j>=5){continue;}
                if(j>=4){
                    i++;
                }
                j = (j+1)%5;
                if(i>=5 || j>=5){break;}
            }else {
                // System.out.println(" already exstis");
                continue;

            }
        }
        // System.out.println();
        // print(key_matrix);
        // System.out.println("i : "+i+"  j: "+j);
        for( int k=0;k<26;k++){
            char ch= (char)('a'+k);
            // System.out.println("ch => "+ch);
            if(!hs.contains(ch)){

                key_matrix[i][j]=ch;
                if(ch=='i' || ch=='j'){
                    get_cordinate.put('i', new Cordinate(i,
j));
                    get_cordinate.put('j', new Cordinate(i,
j));
                    hs.add('i');
                    hs.add('j');
                }else {
```

```
                    get_cordinate.put(ch, new Cordinate(i,
j));
                }

                if(j>=4){
                    i++;
                }
                j = (j+1)%5;
                if(i>=5 || j>=5){
                    break;
                }
                // System.out.println("i => "+i+" j => "+j);

            }
        }
        // System.out.println("------------------");
        print(key_matrix);
    }
    public static ArrayList<String>
generate_valid_digram(String str){
        ArrayList<String> ans=new ArrayList<>();
        int n=str.length();
        ans.add(str.charAt(0)+"");
        for(int i=1;i<n;i++){
            if(ans.get(ans.size()-1).length()==2){
                /*agar jo greater than 1 6e then you need to
add it for sure. */
                String s= str.charAt(i)+"";
                ans.add(s);
            }else {
                /*if it is less than 1 ...then check if the
character is same or not. */
                if(ans.get(ans.size()-1).charAt(0)==
str.charAt(i)){
```

```
                    String new_str= ans.get(ans.size()-1)+
"x";
                    ans.set(ans.size()-1, new_str);
                    ans.add(str.charAt(i)+"");
                }else {
                    String new_str= ans.get(ans.size()-1)+
str.charAt(i);
                    ans.set(ans.size()-1, new_str);
                }
            }
        }
        if(ans.get(ans.size()-1).length()==1){
            String new_str= ans.get(ans.size()-1)+ "x";
            ans.set(ans.size()-1, new_str);
        }

        System.out.println("ans : "+ans);
        return ans;
    }
    public  static String playfair_cipher(String str){
        char ch1=str.charAt(0), ch2=str.charAt(1);

        Cordinate p1=get_cordinate.get(ch1), p2=
get_cordinate.get(ch2);
        int p1_x= p1.x, p1_y = p1.y;
        int p2_x= p2.x, p2_y = p2.y;
        // System.out.println("p1_x: "+p1_x+" p1_y : "+p1_y);
        // System.out.println("p2_x: "+p2_x+" p2_y : "+p2_y);
        String  encypted_str="";
        /*if they are having the same row... */
        if(p1_x == p2_x){
            /*the encrpted characters would be at the right of
them  */
```

```
            char en_ch1= key_matrix[(p1_x)][(p1_y+1)%5];
            char en_ch2= key_matrix[(p2_x)][(p2_y+1)%5];
            encypted_str = en_ch1+ ""+en_ch2;
            return encypted_str;
        }

        /*if they have the same column  */
        if(p1_y == p2_y){
            /*the encrpted characters would be at the right of
them  */
            char en_ch1= key_matrix[(p1_x+1)%5][(p1_y)];
            char en_ch2= key_matrix[(p2_x+1)%5][(p2_y)];
            encypted_str = en_ch1+ ""+en_ch2;
            return encypted_str;
        }
        /*if they are at the corners of the rectangle. */
        /* ***************************SWAPPING KARU 6U.
*****************************/
        char en_ch1= key_matrix[p1_x][p2_y];
        char en_ch2= key_matrix[p2_x][p1_y];
        encypted_str = en_ch1+ ""+en_ch2;
        return encypted_str;

    }
    public  static String playfair_cipher_decrypt(String str){
        char ch1=str.charAt(0), ch2=str.charAt(1);

        Cordinate p1=get_cordinate.get(ch1), p2=
get_cordinate.get(ch2);
        int p1_x= p1.x, p1_y = p1.y;
        int p2_x= p2.x, p2_y = p2.y;
        // System.out.println("p1_x: "+p1_x+" p1_y : "+p1_y);
        // System.out.println("p2_x: "+p2_x+" p2_y : "+p2_y);
        String  encypted_str="";
```

```
        /*if they are having the same row... */
        if(p1_x == p2_x){
            /*the encrpted characters would be at the right of
them  */
            // System.out.println("same row");
            p1_y = (p1_y==0)?4: p1_y-1;
            p2_y = (p2_y==0)?4: p2_y-1;
            char en_ch1= key_matrix[(p1_x)][p1_y];
            char en_ch2= key_matrix[(p2_x)][p2_y];
            encypted_str = en_ch1+ ""+en_ch2;
            return encypted_str;
        }

        /*if they have the same column  */
        if(p1_y == p2_y){
            p1_x = (p1_x==0)?4: p1_x-1;
            p2_x = (p2_x==0)?4: p2_x-1;
            // System.out.println("same col");
            /*the encrpted characters would be at the right of
them  */
            char en_ch1= key_matrix[p1_x][(p1_y)];
            char en_ch2= key_matrix[p2_x][(p1_y)];
            encypted_str = en_ch1+ ""+en_ch2;
            return encypted_str;
        }
        /*if they are at the corners of the rectangle. */
        /* ***************************SWAPPING KARU 6U.
*****************************/
        char en_ch1= key_matrix[p1_x][p2_y];
        char en_ch2= key_matrix[p2_x][p1_y];
        encypted_str = en_ch1+ ""+en_ch2;
        return encypted_str;

    }
```

```
    public static String encryption(String input_text){
        StringBuilder ans=new StringBuilder();

        /*1.arrange string into valid digram.  */
        /*to replace all the spaces. */
        input_text=input_text.replaceAll("\\s", "");
        /*to convert into the lower case */
        input_text=input_text.toLowerCase();
        /*ignore the full stop */
        // System.out.println("input_text: "+input_text);
        ArrayList<String> valid_diagram=
generate_valid_digram(input_text);;

        /*then encrypt it according to the new string... */
        int n=valid_diagram.size();
        for(int i=0;i<n;i++){
            String component= valid_diagram.get(i);
            String encrypted_component=
playfair_cipher(component);
            // System.out.println("component : "+component+"
encrypted_component:  "+encrypted_component);

            ans.append(encrypted_component);
        }
        return ans.toString();
    }
    public static String decryption(String encrypted_text){
        ArrayList<String> valid_diagram=
generate_valid_digram(encrypted_text);;
StringBuilder ans=new StringBuilder();
        /*then encrypt it according to the new string... */
        int n=valid_diagram.size();
        for(int i=0;i<n;i++){
            String component= valid_diagram.get(i);
```

```
            String encrypted_component=
playfair_cipher_decrypt(component);
            // System.out.println("component : "+component+"
encrypted_component:  "+encrypted_component);

            ans.append(encrypted_component);
        }
        return ans.toString();
    }


  public static void main(String[] args) {

    init();
    Scanner sc = new Scanner(System.in);
    /*get the key phrase */
    // String key_phrase = sc.nextLine();
    String key_phrase = "Neso app";

    generate_key_matrix(key_phrase);

    /*enter the plain text */
    // System.out.println("Enter the plain Text");
    // String input_text=sc.nextLine();
    String input_text="youareawesome";
    // String input_text= "odzfqsezsontsw";

    // String input_text= "qs";

    String encrypted_text= encryption(input_text);
    System.out.println("Encrypted text is : "+encrypted_text);

    String decrypted_text= decryption(encrypted_text);
    System.out.println("Decrypted text is : "+decrypted_text);
```

```
    /*THIS IS TO REMOVE THE PADDING */
    if(decrypted_text.charAt(decrypted_text.length()-1)=='x'){

System.out.println(decrypted_text.substring(0,decrypted_text.l
ength()-1));
    }else {
        System.out.println("Decrypted text is :
"+decrypted_text);

    }

    sc.close();
    /*make the key matrix */
  }
}
```

```
C:\Users\Dell\Desktop\study\allStudyMaterial-\sem 6\01_information security\02_labs\04_lab>java p1_playfair_5
_5.java
n  e  s  o  a
p  b  c  d  f
g  h  i  k  l
m  q  r  t  u
v  w  x  y  z
ans : [yo, ua, re, aw, es, om, ex]
Encrypted text is : odzfqsezsontsw
ans : [od, zf, qs, ez, so, nt, sw]
Decrypted text is : youareawesomex
youareawesome
```

## 6 x 6 matrix

```java
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Scanner;

public class p2_playfair_6_6 {
    public static Character key_matrix[][]=new Character[6][6];
    public static void print(Character arr[][]){
        for(int i=0;i<arr.length;i++){
            for(int j=0;j<arr.length;j++){
            System.out.print(arr[i][j]+"  ");
            }
            System.out.println();
        }
    }
    public static class Cordinate{
        int x,y;
        Cordinate(int x,int y){this.x=x;this.y=y;}
    }
    public static HashMap<Character,Cordinate>get_cordinate=new HashMap<>();
    public static void generate_key_matrix(String key_phrase){
        /*keep a track of the characters which have already occured. */
        HashSet<Character> hs=new HashSet<>();
        int n=key_phrase.length(),i=0,j=0;
        key_phrase= key_phrase.toLowerCase();
        // System.out.println("n : "+n+" i: "+i+" j: "+j+" key_phrase"+key_phrase);

        for(int idx=1;idx<=n;idx++){
            char ch=key_phrase.charAt(idx-1);
            // System.out.println("idx: "+idx+" ch: "+ch+" i : "+i+" j: "+j);
            if(!hs.contains(ch)){
                /*ignore if it is a space character */
                if(ch==' '){
                    // System.out.println("----------- space ------");
                    continue;
                }

                key_matrix[i][j]=ch;
                get_cordinate.put(ch, new Cordinate(i, j));
                hs.add(ch);

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

```
                j = (j+1)%6;
                if(i>=6 || j>=6){break;}
            }else {
                // System.out.println(" already exstis");
                continue;

            }
        }
        // print(key_matrix);
        // System.out.println("i : "+i+"  j: "+j);
        for( int k=0;k<n;k++){
            char ch= (char)('a'+k);

            if(!hs.contains(ch)){
                key_matrix[i][j]=ch;
                get_cordinate.put(ch, new Cordinate(i, j));

                if(j>=5){
                    i++;
                }
                j = (j+1)%6;
                if((i)>=6 || (j)>=6){
                    break;
                }
            }
        }
        System.out.println("i : "+i+"  j: "+j);
        i=4;j=2;
        for(int k=0;k<=9;k++){
            char ch= (char)(k+'0');
            key_matrix[i][j]=ch;
            get_cordinate.put(ch, new Cordinate(i, j));
            if(j>=5){
                i++;
            }
            j = (j+1)%6;
            if(i>=6 || j>=6){
                break;
            }
        }
        print(key_matrix);
    }
    public static ArrayList<String> generate_valid_digram(String str){
        ArrayList<String> ans=new ArrayList<>();
        int n=str.length();
        ans.add(str.charAt(0)+"");
        for(int i=1;i<n;i++){
```

```
            if(ans.get(ans.size()-1).length()==2){
                /*agar jo greater than 1 6e then you need to add it for sure. */
                String s= str.charAt(i)+"";
                ans.add(s);
            }else {
                /*if it is less than 1 ...then check if the character is same or not.
*/
                if(ans.get(ans.size()-1).charAt(0)== str.charAt(i)){
                    String new_str= ans.get(ans.size()-1)+ "x";
                    ans.set(ans.size()-1, new_str);
                    ans.add(str.charAt(i)+"");
                }else {
                    String new_str= ans.get(ans.size()-1)+ str.charAt(i);
                    ans.set(ans.size()-1, new_str);
                }
            }
        }
        if(ans.get(ans.size()-1).length()==1){
            String new_str= ans.get(ans.size()-1)+ "x";
            ans.set(ans.size()-1, new_str);
        }

        System.out.println("ans : "+ans);
        return ans;
    }
    public static String playfair_cipher(String str){
        char ch1=str.charAt(0), ch2=str.charAt(1);

        Cordinate p1=get_cordinate.get(ch1), p2= get_cordinate.get(ch2);
        int p1_x= p1.x, p1_y = p1.y;
        int p2_x= p2.x, p2_y = p2.y;
        // System.out.println("p1_x: "+p1_x+" p1_y : "+p1_y);
        // System.out.println("p2_x: "+p2_x+" p2_y : "+p2_y);
        String  encypted_str="";
        /*if they are having the same row... */
        if(p1_x == p2_x){
            // System.out.println("same row");
            /*the encrpted characters would be at the right of them  */
            char en_ch1= key_matrix[(p1_x+1)%6][p1_y];
            char en_ch2= key_matrix[(p2_x+1)%6][p1_y];
            encypted_str = en_ch1+ ""+en_ch2;
            return encypted_str;
        }

        /*if they have the same column  */
        if(p1_y == p2_y){
            // System.out.println("same column");
```

```
            /*the encrpted characters would be at the right of them  */
            char en_ch1= key_matrix[(p1_x+1)%6][(p1_y)];
            char en_ch2= key_matrix[(p2_x+1)%6][(p2_y)];
            encypted_str = en_ch1+ ""+en_ch2;
            return encypted_str;
        }
        /*if they are at the corners of the rectangle. */
        /* *************************SWAPPING KARU 6U. ***************************/
        // System.out.println(" diagonal way");
        char en_ch1= key_matrix[p1_x][p2_y];
        char en_ch2= key_matrix[p2_x][p1_y];
        encypted_str = en_ch1+ ""+en_ch2;
        return encypted_str;

    }
    public static String encryption(String input_text){
        StringBuilder ans=new StringBuilder();

        /*1.arrange string into valid digram.  */
        /*to replace all the spaces. */
        input_text=input_text.replaceAll("\\s", "");
        /*to convert into the lower case */
        input_text=input_text.toLowerCase();
        /*ignore the full stop */
        // System.out.println("input_text: "+input_text);
        ArrayList<String> valid_diagram= generate_valid_digram(input_text);;

        /*then encrypt it according to the new string... */
        int n=valid_diagram.size();
        for(int i=0;i<n;i++){
            String component= valid_diagram.get(i);
            String encrypted_component= playfair_cipher(component);
            // System.out.println("component : "+component+" encrypted_component:
"+encrypted_component);

            ans.append(encrypted_component);
        }
        return ans.toString();
    }

    public  static String playfair_cipher_decrypt(String str){
        char ch1=str.charAt(0), ch2=str.charAt(1);

        Cordinate p1=get_cordinate.get(ch1), p2= get_cordinate.get(ch2);
        int p1_x= p1.x, p1_y = p1.y;
        int p2_x= p2.x, p2_y = p2.y;
        // System.out.println("p1_x: "+p1_x+" p1_y : "+p1_y);
```

```
        // System.out.println("p2_x: "+p2_x+" p2_y : "+p2_y);
        String  encypted_str="";
        /*if they are having the same row... */
        if(p1_x == p2_x){
            /*the encrpted characters would be at the right of them  */
            // System.out.println("same row");
            p1_y = (p1_y==0)?5: p1_y-1;
            p2_y = (p2_y==0)?5: p2_y-1;
            char en_ch1= key_matrix[(p1_x)][p1_y];
            char en_ch2= key_matrix[(p2_x)][p2_y];
            encypted_str = en_ch1+ ""+en_ch2;
            return encypted_str;
        }

        /*if they have the same column  */
        if(p1_y == p2_y){
            p1_x = (p1_x==0)?5: p1_x-1;
            p2_x = (p2_x==0)?5: p2_x-1;
            // System.out.println("same col");
            /*the encrpted characters would be at the right of them  */
            char en_ch1= key_matrix[p1_x][(p1_y)];
            char en_ch2= key_matrix[p2_x][(p1_y)];
            encypted_str = en_ch1+ ""+en_ch2;
            return encypted_str;
        }
        /*if they are at the corners of the rectangle. */
        /* ***************************SWAPPING KARU 6U. ****************************/
        char en_ch1= key_matrix[p1_x][p2_y];
        char en_ch2= key_matrix[p2_x][p1_y];
        encypted_str = en_ch1+ ""+en_ch2;
        return encypted_str;

    }

    public static String decryption(String encrypted_text){
        ArrayList<String> valid_diagram= generate_valid_digram(encrypted_text);;
        StringBuilder ans=new StringBuilder();
                /*then encrypt it according to the new string... */
                int n=valid_diagram.size();
                for(int i=0;i<n;i++){
                    String component= valid_diagram.get(i);
                    String encrypted_component= playfair_cipher_decrypt(component);
                    // System.out.println("component : "+component+"
encrypted_component:  "+encrypted_component);

                    ans.append(encrypted_component);
                }
```

```
                return ans.toString();


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


    Scanner sc = new Scanner(System.in);
    /*get the key phrase */
    // String key_phrase = sc.nextLine();
    String key_phrase = "PLAYFAIR IS A DIGRAM CIPHER";

    generate_key_matrix(key_phrase);

    /*enter the plain text */
    // System.out.println("Enter the plain Text");
    // String input_text=sc.nextLine();
    String input_text= "Hide the gold under the carpet";

    // String input_text= "ox";

    String encrypted_text= encryption(input_text);
    System.out.println("Encrypted text is : "+encrypted_text);

    String decrypted_text= decryption(encrypted_text);
    System.out.println("Decrypted text is : "+decrypted_text);

      /*THIS IS TO REMOVE THE PADDING */
    if(decrypted_text.charAt(decrypted_text.length()-1)=='x'){
        System.out.println(decrypted_text.substring(0,decrypted_text.length()-1));
    }


    sc.close();
    /*make the key matrix */
  }
}
```

```
C:\Users\Dell\Desktop\study\allStudyMaterial-\sem 6\01_information security\02_labs\04_lab>java p2_playfair_6
_6.java
i : 4  j: 3
p  l  a  y  f  i
r  s  d  g  m  c
h  e  b  j  k  n
o  q  t  u  v  w
x  z  0  1  2  3
4  5  6  7  8  9
ans : [hi, de, th, eg, ol, du, nd, er, th, ec, ar, pe, tx]
Encrypted text is : npsbobjsqpgtbchsobnspdlho0
ans : [np, sb, ob, js, qp, gt, bc, hs, ob, ns, pd, lh, o0]
Decrypted text is : hidethegoldunderthecarpetx
hidethegoldunderthecarpet

C:\Users\Dell\Desktop\study\allStudyMaterial-\sem 6\01_information security\02_labs\04_lab>
```