LAB – 3

Aim: Write a program to implement

1. Playfair Cipher
2. Autokey Cipher

1) Playfair Cipher :

```
import java.util.*;

class playfair {
   public static Boolean sameColumn(char[][] matrix, String str) {
      int row1 = 0, row2 = 0;
      int col1 = 0, col2 = 0;

      for (int i = 0; i < 5; i++) {
         for (int j = 0; j < 5; j++) {
            if (matrix[i][j] == str.charAt(0)) {
               row1 = i;
               col1 = j;
            }

            if (matrix[i][j] == str.charAt(1)) {
               row2 = i;
               col2 = j;
            }
         }
      }

      return col1 == col2;
   }

   public static Boolean sameRow(char[][] matrix, String str) {
      int row1 = 0, row2 = 0;
      int col1 = 0, col2 = 0;

      for (int i = 0; i < 5; i++) {
         for (int j = 0; j < 5; j++) {
            if (matrix[i][j] == str.charAt(0)) {
               row1 = i;
               col1 = j;
            }

            if (matrix[i][j] == str.charAt(1)) {
               row2 = i;
```

```
                col2 = j;
            }
        }
    }

    return row1 == row2;
}

public static String encrypt(ArrayList<String> list, char[][] matrix) {
    StringBuilder encrypt = new StringBuilder();

    for (String pair : list) {
        int row1 = 0, row2 = 0;
        int col1 = 0, col2 = 0;

        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++) {
                if (matrix[i][j] == pair.charAt(0)) {
                    row1 = i;
                    col1 = j;
                }

                if (matrix[i][j] == pair.charAt(1)) {
                    row2 = i;
                    col2 = j;
                }
            }
        }

        if (sameRow(matrix, pair)) {
            col1 = (col1 + 1) % 5;
            col2 = (col2 + 1) % 5;
        } else if (sameColumn(matrix, pair)) {
            row1 = (row1 + 1) % 5;
            row2 = (row2 + 1) % 5;
        } else {
            int temp = col1;
            col1 = col2;
            col2 = temp;
        }

        encrypt.append(matrix[row1][col1]).append(matrix[row2][col2]);
    }

    return encrypt.toString();
```

```
}

public static String decrypt(ArrayList<String> list, char[][] matrix) {
    StringBuilder decrypt = new StringBuilder();

    for (String pair : list) {
        int row1 = 0, row2 = 0;
        int col1 = 0, col2 = 0;

        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++) {
                if (matrix[i][j] == pair.charAt(0)) {
                    row1 = i;
                    col1 = j;
                }

                if (matrix[i][j] == pair.charAt(1)) {
                    row2 = i;
                    col2 = j;
                }
            }
        }

        if (sameRow(matrix, pair)) {
            col1 = (col1 - 1 + 5) % 5;
            col2 = (col2 - 1 + 5) % 5;
        } else if (sameColumn(matrix, pair)) {
            row1 = (row1 - 1 + 5) % 5;
            row2 = (row2 - 1 + 5) % 5;
        } else {
            int temp = col1;
            col1 = col2;
            col2 = temp;
        }

        decrypt.append(matrix[row1][col1]).append(matrix[row2][col2]);
    }

    return decrypt.toString();
}

public static String createString (String pText) {
    ArrayList<Character> list = new ArrayList<>();
    StringBuilder ans = new StringBuilder();
```

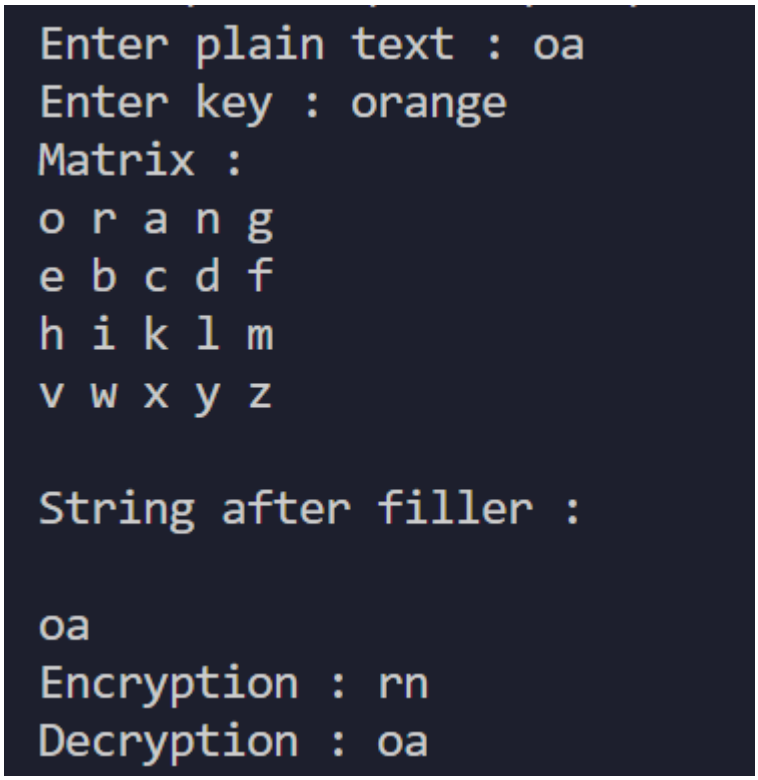
```
        list.add(pText.charAt(0));

        for(int i = 1; i < pText.length(); i++) {
            if(pText.charAt(i - 1) == pText.charAt(i)) {
                list.add('z');
                list.add(pText.charAt(i));
            } else {
                list.add(pText.charAt(i));
            }
        }

        for(int i = 0; i < list.size(); i++) {
            ans.append(list.get(i));
        }

        if ((list.size() ^ 1) == (list.size() + 1)) {

        } else {
            ans.append('z');
        }

        return ans.toString();
    }

    public static ArrayList<String> destructString(String text) {
        ArrayList<String> list = new ArrayList<>();

        for(int i = 0, j = 1; j < text.length(); i += 2, j += 2) {
            StringBuilder temp = new StringBuilder();
            temp.append(text.charAt(i));
            temp.append(text.charAt(j));
            list.add(temp.toString());
        }

        return list;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ArrayList<Character> visited = new ArrayList<>();
        ArrayList<String> destructive = new ArrayList<>();
        ArrayList<String> toDecrypt = new ArrayList<>();

        char[][] matrix = new char[5][5];
        int count = 0;
```

```
System.out.print("Enter plain text : ");
String pText = sc.nextLine();
System.out.print("Enter key : ");
String key = sc.nextLine();

for(int i = 0; i < key.length(); i++) {
    if(!visited.contains(key.charAt(i))) {
        visited.add(key.charAt(i));
    }
}

for(int i = 0; i < 26; i++) {
    char c = (char)('a' + i);
    if(c == 'j') {
        continue;
    }
    if(!visited.contains(c)) {
        visited.add(c);
    }
}

for(int i = 0; i < 5; i++) {
    for(int j = 0; j < 5; j++) {
        matrix[i][j] = visited.get(count);
        count++;
    }
}

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

    System.out.println();
}

System.out.println("\nString after filler : \n");
String fillerString = createString(pText);
System.out.println(fillerString);

destructive = destructString(fillerString);

String encrypted = encrypt(destructive, matrix);
```

```
            toDecrypt = destructString(encrypted);
            String decrypted = decrypt(toDecrypt, matrix);
            System.out.println("Encryption : " + encrypted);
            System.out.println("Decryption : " + decrypted);

            sc.close();
        }
    }
```

OUTPUT :

Row case :



CE019

Column case :

```
Enter plain text : km
Enter key : orange
Matrix :
o r a n g
e b c d f
h i k l m
p q s t u
v w x y z

String after filler :

km
Encryption : lh
Decryption : km
```

Not in same row or column case :

```
Enter plain text : ap
Enter key : orange
Matrix :
o r a n g
e b c d f
h i k l m
v w x y z

String after filler :

ap
Encryption : os
Decryption : ap
```

Consecutive same character case :

```
Enter plain text : hello
Enter key : orange
Matrix :
o r a n g
e b c d f
h i k l m
v w x y z

String after filler :

helzlo
Encryption : phmyhn
Decryption : helzlo
```

Odd length case :

```
Enter plain text : welcome
Enter key : orange
Matrix :
o r a n g
e b c d f
h i k l m
v w x y z

String after filler :

welcomez
Encryption : vbkdghfv
Decryption : welcomez
```

2) Autokey Cipher :

import java.util.*;

```
class auto {
    public static String encrypt (String pText, int key) {
        StringBuilder encrypt = new StringBuilder();

        for(int i = 0; i < pText.length(); i++) {
            if(i == 0) {
                int ascii = ((pText.charAt(i) - 'a') + key) % 26;
                encrypt.append((char)(ascii + 'a'));
            } else {
                int ascii = ((pText.charAt(i) - 'a') + (pText.charAt(i - 1) - 'a')) % 26;
                encrypt.append((char)(ascii + 'a'));
            }
        }

        return encrypt.toString();
    }

    public static String decrypt (String encrypted, int key) {
        StringBuilder decrypt = new StringBuilder();
        int previous = 0;

        for(int i = 0; i < encrypted.length(); i++) {
            if(i == 0) {
                int ascii = ((encrypted.charAt(i) - 'a') - key) % 26;
                previous = ascii;
                decrypt.append((char)(ascii + 'a'));
            } else {
                int ascii = ((encrypted.charAt(i) - 'a') - previous) % 26;

                while(ascii < 0) {
                    ascii += 26;
                }

                previous = ascii;
                decrypt.append((char)(ascii + 'a'));
            }
        }

        return decrypt.toString();
    }

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

        System.out.print("Enter plain text : ");
```

```
        String pText = sc.nextLine();
        System.out.print("Enter key : ");
        int key = sc.nextInt();
        String encrypted = encrypt(pText, key);
        String decrypted = decrypt(encrypted, key);
        System.out.println("Encryption : " + encrypted);
        System.out.println("Decryption : " + decrypted);

        sc.close();
    }
}
```

OUTPUT :