Practical No. 1

Aim:

- **Implementing Substitution and Transposition Ciphers:**
- **Design and implement algorithms to encrypt and decrypt messages using classical**
- **substitution and transposition techniques.**

a) **Caesar Cipher**
Code:

```
import java.io.*;
public class Caesar_cipher {
   public static void main(String args[]) throws IOException {
      System.out.println("caesar cipher");
      System.out.println();
      System.out.println("enter the text");
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      String s1 = new String();
      s1 = br.readLine();
      char c[] = s1.toCharArray();
      int a[] = new int[s1.length()];
      int b[] = new int[s1.length()];
      char d[] = new char[s1.length()];
      for (int i = 0; i < s1.length(); i++) {
         a[i] = (int) c[i];
         b[i] = a[i] + 3;
         d[i] = (char) b[i];
      }
      System.out.println();
      System.out.println("encrypted");
      String s2 = new String(d);
      System.out.println(s2);
      char c1[] = s2.toCharArray();
      int a1[] = new int[s2.length()];
      int b1[] = new int[s2.length()];
      char d1[] = new char[s2.length()];
      for (int i = 0; i < s2.length(); i++) {
         a1[i] = (int) c1[i];
         b1[i] = a1[i] - 3;
         d1[i] = (char) b1[i];
      }
      System.out.println();
      System.out.println("decrypted");
      String s3 = new String(d1);
      System.out.println(s3);
   }}
```

Output:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL                                   Run: Caesar_cipher

PS D:\INS> & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Rushi\AppData\Roaming\Code\User\workspaceStorage\
39fe4244c7c0a81c01c7a368fd6a588a\redhat.java\jdt_ws\INS_775156b5\bin' 'Caesar_cipher'
caesar cipher

enter the text
network

encrypted
qhwzrun

decrypted
network
PS D:\INS>
```

b) Mono-Alphabetic Cipher

Code:

```
import java.io.*;
public class Mono_Alphabetic {
   public static void main(String args[]) throws IOException {
      System.out.println("monoalphabetic cipher");
      System.out.print("enter the text : ");
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      String s1 = new String();
      s1 = br.readLine();
      char userText[] = s1.toCharArray();
      char normal[] = { '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' };
      char cipher[] = { 'r', 'i', 't', 'e', 's', 'h', 'v', 'w', 'a', 'k', 'm', 'y', 'p', 'o', 'j', 'f', 'x', 'q', 'x',
            'd', 'g', 'z', 'c', 'n', 'l', 'b' };
      char Encry[] = new char[cipher.length];
      char Decry[] = new char[cipher.length];
      int j = 0;
      for (int i = 0; i < userText.length; i++)
         for (j = 0; j < 26; j++)
            if (normal[j] == userText[i])
               Encry[i] = cipher[j];
      for (int i = 0; i < userText.length; i++)
         for (j = 0; j < 26; j++)
            if (cipher[j] == Encry[i])
               Decry[i] = normal[j];
      System.out.println();
      for (int i = 0; i < userText.length; i++)
         System.out.print(userText[i]);
      System.out.println(" -> Your Input Plain(Original) Text");
      System.out.println();
      for (int i = 0; i < userText.length; i++)
         System.out.print(Encry[i]);
      System.out.println(" -> After Encrypted Text");
      System.out.println();
      for (int i = 0; i < userText.length; i++)
         System.out.print(Decry[i]);
      System.out.println(" -> After Decryption Text");
   }
}
```

Output:

```
PS D:\INS>  & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Rushi\AppData\Roaming\Code\User\workspaceStorage\
39fe4244c7c0a81c01c7a368fd6a588a\redhat.java\jdt_ws\INS_775156b5\bin' 'Mono_Alphabetic'
monoalphabetic cipher
enter the text : network

network -> Your Input Plain(Original) Text

osdcjqm -> After Encrypted Text

network -> After Decryption Text
PS D:\INS>
```

c) Poly-Alphabetic Cipher
Code:

```
import java.io.*;
public class Polyalphabetic {
   public static void main(String args[]) throws IOException {
      int t = 0;
      int u = 0;
      System.out.println("Polyalphabetic cipher");
      System.out.print("enter the text : ");
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      String s1 = new String();
      s1 = br.readLine();
      char m[] = s1.toCharArray();
      System.out.print("enter the key : ");
      String s2 = new String();
      s2 = br.readLine();
      char kk[] = s2.toCharArray();
      char k[] = new char[m.length];
      char ct[] = new char[m.length];
      char pt[] = new char[m.length];
      char mat[][] = new char[26][26];
      int l = 97;
      int d = 0;
      int w = 0;
      for (int i = 0; i < s1.length(); i++) {
         k[i] = kk[i % s2.length()];
      }
      for (int i = 0; i < 26; i++) {
         l = l + i;
         for (int j = 0; j < 26; j++) {
            d = l + j;
            if (d > 122) {
               mat[i][j] = (char) (d - 26);
            } else {
               mat[i][j] = (char) d;
            }
         }
         d = 0;
         l = 97;
      }
      for (int i = 0; i < 26; i++) {
         for (int j = 0; j < 26; j++) {
            System.out.print(mat[i][j] + " ");
         }
         System.out.println();
      }
      // --------encryption--------
      for (int q = 0; q < m.length; q++) {
```

```
            for (int i = 0; i < 26; i++) {
                int j = 0;
                if (mat[i][j] == k[q]) // finding the row corresponding to key
                {
                    t = i;
                }
            }
            for (int i = 0; i < 26; i++) {
                int j = 0;
                if (mat[j][i] == m[q]) // finding the column according to text
                {
                    u = i;
                }
            }
            if (m[q] == ' ')
                ct[q] = '`';
            else
                ct[q] = mat[t][u]; // intersection of row key and column text gives cipher text
        }
        System.out.print(ct);
        System.out.println(" -> Cipher text");
        // -----------decryption------
        for (int q = 0; q < m.length; q++) {
            for (int i = 0; i < 26; i++) {
                int j = 0;
                if (mat[i][j] == k[q]) // finding the row corresponding to key
                {
                    t = i;
                }
            }
            for (int i = 0; i < 26; i++) {
                int j = t;
                if (mat[j][i] == ct[q])// finding the cipher text in row corresponding to key
                {
                    u = i;
                }
            }
            if (ct[q] == '`')
                pt[q] = ' ';
            else
                pt[q] = mat[0][u];// intersection of zeroth row and column corresponding to cipher text
position
        }
        System.out.print(pt);
        System.out.println(" -> Plain text");
    }
}
```

Output:

```
PS D:\INS>  & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Rushi\AppData\Roaming\Code\User\workspaceStorage\
39fe4244c7c0a81c01c7a368fd6a588a\redhat.java\jdt_ws\INS_775156b5\bin' 'Polyalphabetic'
Polyalphabetic cipher
enter the text : network
enter the key : secret
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
fivnskc -> Cipher text
network -> Plain text
PS D:\INS>
```

d) Vernam Cipher

Code:

```
import java.io.*;

public class vernam {
   public static void main(String a[]) throws IOException {
      String text = new String("HELLO");
      char[] arText = text.toCharArray();
      String Cipher = new String("XYZHG");
      char[] arCipher = Cipher.toCharArray();
      char[] encoded = new char[5];
      System.out.println("Encrypted test");
      for (int i = 0; i < arText.length; i++) {
         encoded[i] = (char) (arText[i] ^ arCipher[i]); // XOR operation
         System.out.println(encoded[i]);
      }
      System.out.println("Decrypted text:");
      for (int i = 0; i < encoded.length; i++) {
         char temp = (char) (encoded[i] ^ arCipher[i]);
         System.out.println(temp);
      }
   }
}
```

Output:

```
PS D:\INS> & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Rushi\AppData\Roaming\Code\User\workspaceStorage\
39fe4244c7c0a81c01c7a368fd6a588a\redhat.java\jdt_ws\INS_775156b5\bin' 'vernam'
Encrypted test
►
L
▪
♦

Decrypted text:
H
E
L
L
O
PS D:\INS>
```

e) Playfair Cipher
Code:

```
import java.io.*;
class PlayFair {
    String key = new String();
    String key2 = new String();
    String text = new String();
    char key_array[][] = new char[5][5];
    public void keySetter(String k) {
        String str = new String();
        boolean test = false;
        str = str + k.charAt(0);
        for (int i = 1; i < k.length(); i++) {
            for (int j = 0; j < str.length(); j++)
                if (k.charAt(i) == str.charAt(j) || k.charAt(i) == 'j')
                    test = true;
            if (!test)
                str = str + k.charAt(i);
            test = false;
        }
        key = str;
        matrixBuilder(key);
    }
    public void matrixBuilder(String k) {
        key2 = key2 + key;
        boolean test = false;
        char current;
        for (int i = 0; i < 26; i++) {
            current = (char) (i + 97);
            for (int j = 0; j < key.length(); j++)
                if (current == 'j' || current == key.charAt(j))
                    test = true;
            if (!test)
                key2 = key2 + current;
            test = false;
        }
        System.out.println(key2);
        int a = 0;
        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++) {
                key_array[i][j] = key2.charAt(a);
                a++;
            }
        }
        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++)
                System.out.print(key_array[i][j] + " ");
            System.out.println();
```

```
        }
    }
    public void stringConversion(String input) {
        String altered = new String();
        altered = input.replace('j', 'i');
        for (int i = 0; i < altered.length(); i++)
            if (i > 0 && altered.charAt(i) == altered.charAt(i - 1))
                altered = altered.substring(0, i) + 'x' + altered.substring(i);
        if ((altered.length() % 2) != 0)
            altered = altered + 'x';
        text = altered;
        System.out.println(text);
    }
    public int[] getDimensions(char letter) {
        int key[] = new int[2];
        for (int i = 0; i < 5; i++)
            for (int j = 0; j < 5; j++)
                if (key_array[i][j] == letter) {
                    key[0] = i; // row index position of character in matrix
                    key[1] = j; // column index position of character in matrix
                    break;
                }
        return key;
    }
    public void Encrypt() {
        char a, b;
        String Code = "";
        int c[] = new int[2];
        int d[] = new int[2];
        for (int i = 0; i < text.length(); i = i + 2) // Plain Text groups
        {
            a = text.charAt(i); // First PT character
            b = text.charAt(i + 1); // Second PT character
            c = getDimensions(a);
            d = getDimensions(b);
            if (c[0] == d[0])// Same Row
            {
                if (c[1] < 4)
                    c[1]++; // replaced with character right side
                else
                    c[1] = 0; // matrix wrap
                if (d[1] < 4)
                    d[1]++; // replaced with character right side
                else
                    d[1] = 0; // matrix wrap
            } else if (c[1] == d[1]) {
                if (c[0] < 4)
                    c[0]++;
                else
```

```
                c[0] = 0;
            if (d[0] < 4)
                d[0]++;
            else
                d[0] = 0;
        } else {
            int temp = c[1];
            c[1] = d[1];
            d[1] = temp;
        }
        Code = Code + key_array[c[0]][c[1]] + key_array[d[0]][d[1]];
    }
    System.out.println("Encrypted text:" + Code);
}

public void Decrypt() {
    char a, b;
    String Code = "";
    int c[] = new int[2];
    int d[] = new int[2];
    for (int i = 0; i < text.length(); i = i + 2) {
        a = text.charAt(i);
        b = text.charAt(i + 1);
        c = getDimensions(a);
        d = getDimensions(b);
        if (c[0] == d[0]) {
            if (c[1] > 0)
                c[1]--;
            else
                c[1] = 4;
            if (d[1] > 0)
                d[1]--;
            else
                d[1] = 4;
        } else if (c[1] == d[1]) {
            if (c[0] > 0)
                c[0]--;
            else
                c[0] = 4;
            if (d[0] > 0)
                d[0]--;
            else
                d[0] = 4;
        } else {
            int temp = c[1];
            c[1] = d[1];
            d[1] = temp;
        }
        Code = Code + key_array[c[0]][c[1]] + key_array[d[0]][d[1]];
```

```
        }
        System.out.println("Decrypted text:" + Code);
    }

    public static void main(String args[]) throws IOException {
        String s, s2, s3, s4;
        int ch;
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        PlayFair p = new PlayFair();
        System.out.println("Enter the key:");
        s4 = br.readLine();
        p.keySetter(s4);
        do {
            System.out.println("Enter Your Choice:\n1.Encrypt\n2.Decrypt\n3.Exit");
            s = br.readLine();
            ch = Integer.valueOf(s).intValue();
            switch (ch) {
                case 1:
                    System.out.println("Enter Text to be Encrypted:");
                    s2 = br.readLine();
                    s2 = s2.toLowerCase();
                    p.stringConversion(s2);
                    p.Encrypt();
                    break;
                case 2:
                    System.out.println("Enter Text to be Decrypted:");
                    s3 = br.readLine();
                    s3 = s3.toLowerCase();
                    p.stringConversion(s3);
                    p.Decrypt();
                    break;
                case 3:
                    System.exit(0);
            }
        } while (ch < 3);
    }
}
```

Output:

```
PS D:\INS>  & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Rushi\AppData\Roaming\Code\User\workspaceStorage\
39fe4244c7c0a81c01c7a368fd6a588a\redhat.java\jdt_ws\INS_775156b5\bin' 'PlayFair'
Enter the key:
work
workabcdefghilmnpqstuvxyz
w o r k a
b c d e f
g h i l m
n p q s t
u v x y z
Enter Your Choice:
1.Encrypt
2.Decrypt
3.Exit
1
Enter Text to be Encrypted:
hell
helxlx
Encrypted text:lciyiy
Enter Your Choice:
1.Encrypt
2.Decrypt
3.Exit
2
Enter Text to be Decrypted:
lciyiy
lciyiy
Decrypted text:helxlx
Enter Your Choice:
1.Encrypt
2.Decrypt
3.Exit
3
PS D:\INS>
```

f) Rail Fence

Code:

```
import java.io.*;
public class railfence {
    public static void main(String args[]) throws IOException {
        System.out.println("Enter plain text:");
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String t = br.readLine();
        String lpt = "", rpt = "", ct = "";
        for (int i = 0; i < t.length(); i++) {
            if (i % 2 == 0)
                lpt += t.charAt(i);
            else
                rpt += t.charAt(i);
        }
        ct = lpt.concat(rpt);
        System.out.println(ct);
        if (ct.length() % 2 == 0) {
        } else {
            ct = ct + " ";
        }
        int a = 0;
        String l = ct.substring(0, ct.length() / 2);
        String r = ct.substring(ct.length() / 2, ct.length());
        char[] pt1 = new char[l.length()];
        char[] pt2 = new char[r.length()];
        for (int i = 0; i < l.length(); i++) {
            pt1[a] = l.charAt(i);
            pt2[a] = r.charAt(i);
            a++;
            System.out.print(pt1[i] + "" + pt2[i]);
        }
    }
}
```

Output:

```
PS D:\INS> & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Rushi\AppData\Roaming\Code\User\
workspaceStorage\39fe4244c7c0a81c01c7a368fd6a588a\redhat.java\jdt_ws\INS_775156b5\bin' 'railfence'
Enter plain text:
network
ntokewr
network
PS D:\INS>
```

g) Simple Columnar
Code:

```
import java.util.*;
class SimpleColumnar {
   public static void main(String sap[]) {
      Scanner sc = new Scanner(System.in);
      System.out.print("\nEnter plaintext(enter in lower case): ");
      String message = sc.next();
      System.out.print("\nEnter key in numbers: ");
      String key = sc.next();
      int columnCount = key.length();
      int rowCount = (message.length() + columnCount) / columnCount;
      int plainText[][] = new int[rowCount][columnCount];
      int cipherText[][] = new int[rowCount][columnCount];
      System.out.print("\n-----Encryption-----\n");
      cipherText = encrypt(plainText, cipherText, message, rowCount, columnCount, key);
      String ct = "";
      for (int i = 0; i < columnCount; i++) {
         for (int j = 0; j < rowCount; j++) {
            if (cipherText[j][i] == 0)
               ct = ct + 'x';
            else {
               ct = ct + (char) cipherText[j][i];
            }
         }
      }
      System.out.print("\nCipher Text: " + ct);
      System.out.print("\n\n\n-----Decryption-----\n");
      plainText = decrypt(plainText, cipherText, ct, rowCount, columnCount, key);
      String pt = "";
      for (int i = 0; i < rowCount; i++) {
         for (int j = 0; j < columnCount; j++) {
            if (plainText[i][j] == 0)
               pt = pt + "";
            else {
               pt = pt + (char) plainText[i][j];
            }
         }
      }
      System.out.print("\nPlain Text: " + pt);
      System.out.println();
   }
   static int[][] encrypt(int plainText[][], int cipherText[][], String message, int rowCount, int
columnCount,
         String key) {
      int i, j;
      int k = 0;
      for (i = 0; i < rowCount; i++) {
```

```
            for (j = 0; j < columnCount; j++) {
                if (k < message.length()) {
                    plainText[i][j] = (int) message.charAt(k);
                    k++;
                } else {
                    break;
                }
            }
        }
        for (i = 0; i < columnCount; i++) {
            int currentCol = ((int) key.charAt(i) - 48) - 1;
            for (j = 0; j < rowCount; j++) {
                cipherText[j][i] = plainText[j][currentCol];
            }
        }
        System.out.print("Cipher Array(read column by column): \n");
        for (i = 0; i < rowCount; i++) {
            for (j = 0; j < columnCount; j++) {
                System.out.print((char) cipherText[i][j] + "\t");
            }
            System.out.println();
        }
        return cipherText;
    }
    static int[][] decrypt(int plainText[][], int cipherText[][], String message, int rowCount, int
columnCount,
            String key) {
        int i, j;
        int k = 0;
        for (i = 0; i < columnCount; i++) {
            int currentCol = ((int) key.charAt(i) - 48) - 1;
            for (j = 0; j < rowCount; j++) {
                plainText[j][currentCol] = cipherText[j][i];
            }
        }
        System.out.print("Plain Array(read row by row): \n");
        for (i = 0; i < rowCount; i++) {
            for (j = 0; j < columnCount; j++) {
                System.out.print((char) plainText[i][j] + "\t");
            }
            System.out.println();
        }
        return plainText;
    }
}
```

Output:

```
PS D:\INS>  & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Rushi\AppData\Roaming\Code\User\workspaceStorage\
39fe4244c7c0a81c01c7a368fd6a588a\redhat.java\jdt_ws\INS_775156b5\bin' 'SimpleColumnar'

Enter plaintext(enter in lower case): secretmessage

Enter key in numbers: 3142

-----Encryption-----
Cipher Array(read column by column):
c       s       r       e
m       e       e       t
a       s       g       s
        e

Cipher Text: cmaxseseregxetsx


-----Decryption-----
Plain Array(read row by row):
s       e       c       r
e       t       m       e
s       s       a       g
e

Plain Text: secretmessage
PS D:\INS>
```