PRACTICAL: 01

AIM : Write programs to implement the following Substitution Cipher Techniques:

A) Caesar Cipher

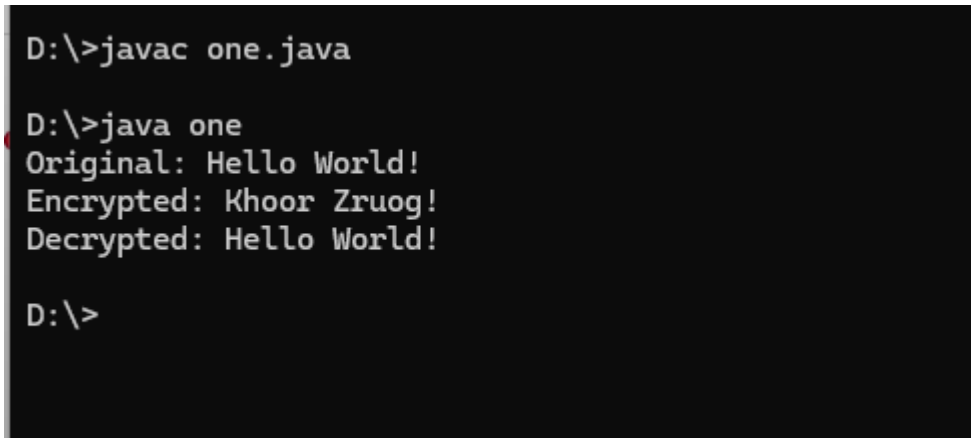
B) Monoalphabetic Cipher

A) Caesar Cipher

```
public class caesarcipher {


   public static String encrypt(String text, int shift) {
      StringBuilder result = new StringBuilder();


      for (int i = 0; i < text.length(); i++) {
         char ch = text.charAt(i);
         if (Character.isLetter(ch)) {
            char base = Character.isLowerCase(ch) ? 'a' : 'A';
            ch = (char) ((ch - base + shift) % 26 + base);
         }
         result.append(ch);
      }


      return result.toString();
   }


   public static String decrypt(String text, int shift) {
      return encrypt(text, 26 - shift);
   }


   public static void main(String[] args) {
      String text = "Hello World!";
      int shift = 3;
```

```
        String encrypted = encrypt(text, shift);
        String decrypted = decrypt(encrypted, shift);

        System.out.println("Original: " + text);
        System.out.println("Encrypted: " + encrypted);
        System.out.println("Decrypted: " + decrypted);
    }
}
```

OUTPUT:

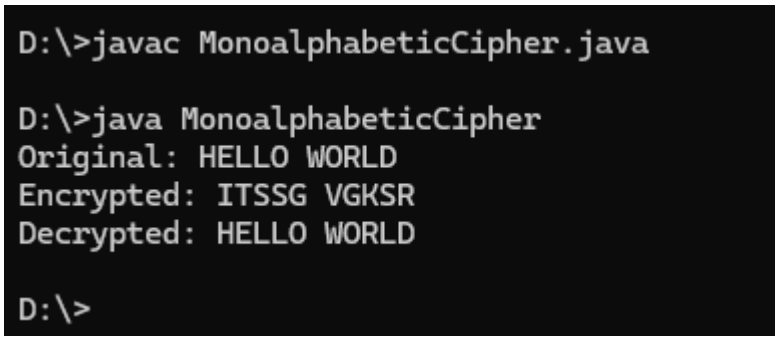


B) Monoalphabetic Cipher

```
import java.util.HashMap;
import java.util.Map;

public class MonoalphabeticCipher {

    private static final String ALPHABET = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

    public static String generateKey() {
        // Simple key for demonstration. For a more secure key, you should randomize the characters.
        String key = "QWERTYUIOPASDFGHJKLZXCVBNM";
        return key;
    }

    public static String encrypt(String text, String key) {
        Map<Character, Character> charMap = createCharMap(ALPHABET, key);
        return transformText(text, charMap);
    }

    public static String decrypt(String text, String key) {
        Map<Character, Character> charMap = createCharMap(key, ALPHABET);
        return transformText(text, charMap);
    }

    private static Map<Character, Character> createCharMap(String from, String to) {
        Map<Character, Character> charMap = new HashMap<>();
        for (int i = 0; i < from.length(); i++) {
            charMap.put(from.charAt(i), to.charAt(i));
        }
        return charMap;
```

```
    }

    private static String transformText(String text, Map<Character, Character> charMap) {
        StringBuilder result = new StringBuilder();

        for (char ch : text.toUpperCase().toCharArray()) {
            if (charMap.containsKey(ch)) {
                result.append(charMap.get(ch));
            } else {
                result.append(ch); // non-alphabetic characters remain unchanged
            }
        }

        return result.toString();
    }

    public static void main(String[] args) {
        String text = "HELLO WORLD";
        String key = generateKey();

        String encrypted = encrypt(text, key);
        String decrypted = decrypt(encrypted, key);

        System.out.println("Original: " + text);
        System.out.println("Encrypted: " + encrypted);
        System.out.println("Decrypted: " + decrypted);
    }
}
```

OUTPUT:


D:\>javac MonoalphabeticCipher.java
D:\>java MonoalphabeticCipher
Original: HELLO WORLD
Encrypted: ITSSG VGKSR
Decrypted: HELLO WORLD
D:\>

PRACTICAL : 02

AIM : Write programs to implement the following Substitution Cipher Techniques:

A) Vernam Cipher

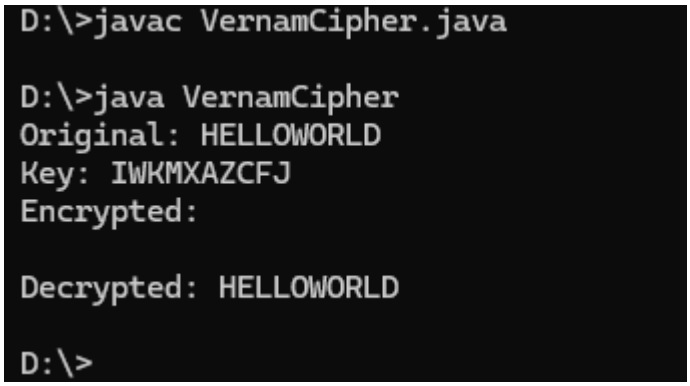
B) Playfair Cipher

A) Vernam Cipher

```
import java.util.Random;

public class VernamCipher {

    public static String generateKey(int length) {
        Random random = new Random();
        StringBuilder key = new StringBuilder();

        for (int i = 0; i < length; i++) {
            char ch = (char) (random.nextInt(26) + 'A');
            key.append(ch);
        }

        return key.toString();
    }

    public static String encrypt(String text, String key) {
        StringBuilder result = new StringBuilder();

        for (int i = 0; i < text.length(); i++) {
            char ch = (char) (text.charAt(i) ^ key.charAt(i));
            result.append(ch);
        }
```

```
        return result.toString();
    }

    public static String decrypt(String text, String key) {
        return encrypt(text, key);  // Encryption and decryption are the same for Vernam Cipher
    }

    public static void main(String[] args) {
        String text = "HELLOWORLD";
        String key = generateKey(text.length());

        String encrypted = encrypt(text, key);
        String decrypted = decrypt(encrypted, key);

        System.out.println("Original: " + text);
        System.out.println("Key: " + key);
        System.out.println("Encrypted: " + encrypted);
        System.out.println("Decrypted: " + decrypted);
    }
}
```

OUTPUT:



B) Playfair Cipher

```
import java.util.HashSet;
import java.util.Set;

public class PlayfairCipher {

    private static char[][] matrix = new char[5][5];
    private static final String ALPHABET = "ABCDEFGHIKLMNOPQRSTUVWXYZ";

    public static void generateKeyMatrix(String key) {
        Set<Character> usedChars = new HashSet<>();
        key = key.toUpperCase().replaceAll("J", "I");
        StringBuilder keyBuilder = new StringBuilder(key);

        for (char ch : ALPHABET.toCharArray()) {
            if (!usedChars.contains(ch) && keyBuilder.indexOf(String.valueOf(ch)) == -1) {
                keyBuilder.append(ch);
            }
        }

        int k = 0;
        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++) {
                char ch = keyBuilder.charAt(k++);
                matrix[i][j] = ch;
```

```
            usedChars.add(ch);
        }
    }
}

public static String preprocessText(String text) {
    text = text.toUpperCase().replaceAll("[^A-Z]", "").replaceAll("J", "I");
    StringBuilder processed = new StringBuilder();

    for (int i = 0; i < text.length(); i++) {
        char ch = text.charAt(i);
        processed.append(ch);
        if (i < text.length() - 1 && text.charAt(i) == text.charAt(i + 1)) {
            processed.append('X');
        }
    }

    if (processed.length() % 2 != 0) {
        processed.append('X');
    }

    return processed.toString();
}

public static String encrypt(String text, String key) {
    generateKeyMatrix(key);
    text = preprocessText(text);
    StringBuilder result = new StringBuilder();

    for (int i = 0; i < text.length(); i += 2) {
        char a = text.charAt(i);
```

```
        char b = text.charAt(i + 1);
        int[] posA = findPosition(a);
        int[] posB = findPosition(b);

        if (posA[0] == posB[0]) {
            result.append(matrix[posA[0]][(posA[1] + 1) % 5]);
            result.append(matrix[posB[0]][(posB[1] + 1) % 5]);
        } else if (posA[1] == posB[1]) {
            result.append(matrix[(posA[0] + 1) % 5][posA[1]]);
            result.append(matrix[(posB[0] + 1) % 5][posB[1]]);
        } else {
            result.append(matrix[posA[0]][posB[1]]);
            result.append(matrix[posB[0]][posA[1]]);
        }
    }

    return result.toString();
}

public static String decrypt(String text, String key) {
    generateKeyMatrix(key);
    StringBuilder result = new StringBuilder();

    for (int i = 0; i < text.length(); i += 2) {
        char a = text.charAt(i);
        char b = text.charAt(i + 1);
        int[] posA = findPosition(a);
        int[] posB = findPosition(b);

        if (posA[0] == posB[0]) {
            result.append(matrix[posA[0]][(posA[1] + 4) % 5]);
```

```
            result.append(matrix[posB[0]][(posB[1] + 4) % 5]);
        } else if (posA[1] == posB[1]) {
            result.append(matrix[(posA[0] + 4) % 5][posA[1]]);
            result.append(matrix[(posB[0] + 4) % 5][posB[1]]);
        } else {
            result.append(matrix[posA[0]][posB[1]]);
            result.append(matrix[posB[0]][posA[1]]);
        }
    }

    return result.toString();
}

private static int[] findPosition(char ch) {
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 5; j++) {
            if (matrix[i][j] == ch) {
                return new int[] { i, j };
            }
        }
    }
    return null;
}

public static void main(String[] args) {
    String text = "HELLO WORLD";
    String key = "KEYWORD";

    String encrypted = encrypt(text, key);
    String decrypted = decrypt(encrypted, key);
```

```
        System.out.println("Original: " + text);
        System.out.println("Key: " + key);
        System.out.println("Encrypted: " + encrypted);
        System.out.println("Decrypted: " + decrypted);
    }
}
```

OUTPUT:

PRACTICAL : 03

AIM : Write programs to implement the following Transposition Cipher Techniques:

A) Rail Fence Cipher.

B) Simple Columnar Technique.

A) Rail Fence Cipher.

```
public class RailFenceCipher {


    public static String encrypt(String text, int key) {
        if (key == 1) return text;


        StringBuilder[] rail = new StringBuilder[key];
        for (int i = 0; i < key; i++) {
            rail[i] = new StringBuilder();
        }


        int row = 0;
        boolean down = true;


        for (char ch : text.toCharArray()) {
            rail[row].append(ch);
            if (row == 0) {
                down = true;
            } else if (row == key - 1) {
                down = false;
            }
            row += down ? 1 : -1;
        }
```

```
    StringBuilder result = new StringBuilder();
    for (StringBuilder sb : rail) {
        result.append(sb);
    }

    return result.toString();
}

public static String decrypt(String text, int key) {
    if (key == 1) return text;

    char[] decrypted = new char[text.length()];
    boolean[] visited = new boolean[text.length()];

    int index = 0;
    for (int k = 0; k < key; k++) {
        int row = 0;
        boolean down = true;

        for (int i = 0; i < text.length(); i++) {
            if (row == k && !visited[i]) {
                decrypted[i] = text.charAt(index++);
                visited[i] = true;
            }
            if (row == 0) {
                down = true;
            } else if (row == key - 1) {
                down = false;
            }
            row += down ? 1 : -1;
        }
```

```
        }

        return new String(decrypted);
    }

    public static void main(String[] args) {
        String text = "HELLO WORLD";
        int key = 3;

        String encrypted = encrypt(text, key);
        String decrypted = decrypt(encrypted, key);

        System.out.println("Original: " + text);
        System.out.println("Encrypted: " + encrypted);
        System.out.println("Decrypted: " + decrypted);
    }
}
```

OUTPUT:

```
D:\>javac RailFenceCipher.java

D:\>java RailFenceCipher
Original: HELLO WORLD
Encrypted: HOREL OLLWD
Decrypted: HELLO WORLD

D:\>
```

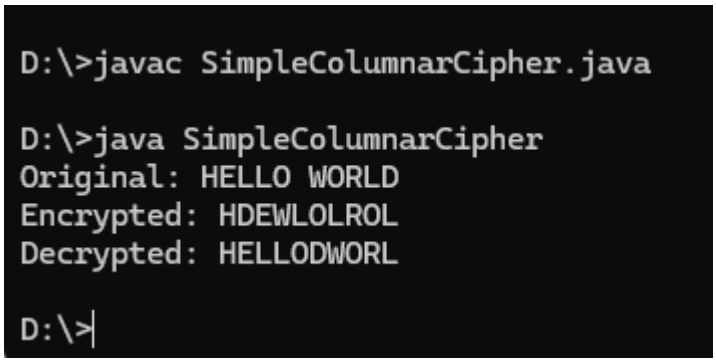
B) Simple Columnar Technique.

```
import java.util.Arrays;

public class SimpleColumnarCipher {

    public static String encrypt(String text, int key) {
        int length = text.length();
        int numRows = (int) Math.ceil((double) length / key);
        char[][] grid = new char[numRows][key];

        for (char[] row : grid) {
            Arrays.fill(row, ' ');
        }

        int index = 0;
        for (int r = 0; r < numRows; r++) {
            for (int c = 0; c < key; c++) {
                if (index < length) {
                    grid[r][c] = text.charAt(index++);
                }
            }
        }

        StringBuilder result = new StringBuilder();
        for (int c = 0; c < key; c++) {
            for (int r = 0; r < numRows; r++) {
                if (grid[r][c] != ' ') {
                    result.append(grid[r][c]);
```

```
            }
        }
    }

    return result.toString();
}

public static String decrypt(String text, int key) {
    int length = text.length();
    int numRows = (int) Math.ceil((double) length / key);
    char[][] grid = new char[numRows][key];

    for (char[] row : grid) {
        Arrays.fill(row, ' ');
    }

    int index = 0;
    for (int c = 0; c < key; c++) {
        for (int r = 0; r < numRows; r++) {
            if (index < length) {
                grid[r][c] = text.charAt(index++);
            }
        }
    }

    StringBuilder result = new StringBuilder();
    for (int r = 0; r < numRows; r++) {
        for (int c = 0; c < key; c++) {
            if (grid[r][c] != ' ') {
                result.append(grid[r][c]);
            }
```

```
            }
        }

        return result.toString();
    }

    public static void main(String[] args) {
        String text = "HELLO WORLD";
        int key = 5;

        String encrypted = encrypt(text, key);
        String decrypted = decrypt(encrypted, key);

        System.out.println("Original: " + text);
        System.out.println("Encrypted: " + encrypted);
        System.out.println("Decrypted: " + decrypted);
    }
}
```

OUTPUT:



PRACTICAL : 04

AIM : Write program to encrypt and decrypt strings using

A) DES Algorithm.

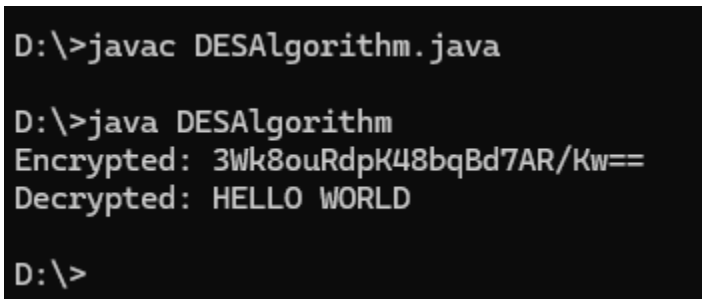
B) AES Algorithm.

A) DES Algorithm.

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.util.Base64;

public class DESAlgorithm {

   public static String encrypt(String plainText, SecretKey secretKey) throws Exception {
      Cipher cipher = Cipher.getInstance("DES");
      cipher.init(Cipher.ENCRYPT_MODE, secretKey);
      byte[] encryptedBytes = cipher.doFinal(plainText.getBytes());
      return Base64.getEncoder().encodeToString(encryptedBytes);
   }

   public static String decrypt(String encryptedText, SecretKey secretKey) throws Exception {
      Cipher cipher = Cipher.getInstance("DES");
      cipher.init(Cipher.DECRYPT_MODE, secretKey);
      byte[] decodedBytes = Base64.getDecoder().decode(encryptedText);
      byte[] decryptedBytes = cipher.doFinal(decodedBytes);
      return new String(decryptedBytes);
   }
```

```
    public static void main(String[] args) throws Exception {
        String plainText = "HELLO WORLD";

        // Generate a DES key
        KeyGenerator keyGenerator = KeyGenerator.getInstance("DES");
        SecretKey secretKey = keyGenerator.generateKey();

        // Encrypt the plaintext
        String encryptedText = encrypt(plainText, secretKey);
        System.out.println("Encrypted: " + encryptedText);

        // Decrypt the encrypted text
        String decryptedText = decrypt(encryptedText, secretKey);
        System.out.println("Decrypted: " + decryptedText);
    }
}
```

OUTPUT:



B) AES Algorithm.

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.util.Base64;

public class AESAlgorithm {

    public static String encrypt(String plainText, SecretKey secretKey) throws Exception {
        Cipher cipher = Cipher.getInstance("AES");
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);
        byte[] encryptedBytes = cipher.doFinal(plainText.getBytes());
        return Base64.getEncoder().encodeToString(encryptedBytes);
    }

    public static String decrypt(String encryptedText, SecretKey secretKey) throws Exception {
        Cipher cipher = Cipher.getInstance("AES");
        cipher.init(Cipher.DECRYPT_MODE, secretKey);
        byte[] decodedBytes = Base64.getDecoder().decode(encryptedText);
        byte[] decryptedBytes = cipher.doFinal(decodedBytes);
        return new String(decryptedBytes);
    }

    public static void main(String[] args) throws Exception {
        String plainText = "HELLO WORLD";

        // Generate an AES key
        KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
        keyGenerator.init(128); // Key size can be 128, 192, or 256 bits
        SecretKey secretKey = keyGenerator.generateKey();
```

```
        // Encrypt the plaintext
        String encryptedText = encrypt(plainText, secretKey);
        System.out.println("Encrypted: " + encryptedText);

        // Decrypt the encrypted text
        String decryptedText = decrypt(encryptedText, secretKey);
        System.out.println("Decrypted: " + decryptedText);
    }
}
```

OUTPUT:

PRACTICAL : 05

AIM : Write a program to implement RSA algorithm to perform encryption / decryption of a given string.

CODE:

```
import java.security.*;
import javax.crypto.Cipher;
import java.util.Base64;

public class RSAAlgorithm {

    public static KeyPair generateKeyPair() throws NoSuchAlgorithmException {
        KeyPairGenerator keyGen = KeyPairGenerator.getInstance("RSA");
        keyGen.initialize(2048); // You can use 1024, 2048, or 4096 bits for stronger security
        return keyGen.genKeyPair();
    }

    public static String encrypt(String plainText, PublicKey publicKey) throws Exception {
        Cipher cipher = Cipher.getInstance("RSA");
        cipher.init(Cipher.ENCRYPT_MODE, publicKey);
        byte[] encryptedBytes = cipher.doFinal(plainText.getBytes());
        return Base64.getEncoder().encodeToString(encryptedBytes);
    }

    public static String decrypt(String encryptedText, PrivateKey privateKey) throws Exception {
        Cipher cipher = Cipher.getInstance("RSA");
        cipher.init(Cipher.DECRYPT_MODE, privateKey);
        byte[] decryptedBytes = cipher.doFinal(Base64.getDecoder().decode(encryptedText));
        return new String(decryptedBytes);
    }
```

```
    public static void main(String[] args) {
        try {
            // Generate RSA key pair
            KeyPair keyPair = generateKeyPair();
            PublicKey publicKey = keyPair.getPublic();
            PrivateKey privateKey = keyPair.getPrivate();

            // Plain text
            String plainText = "HELLO WORLD";

            // Encrypt the plain text
            String encryptedText = encrypt(plainText, publicKey);
            System.out.println("Encrypted: " + encryptedText);

            // Decrypt the encrypted text
            String decryptedText = decrypt(encryptedText, privateKey);
            System.out.println("Decrypted: " + decryptedText);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

OUTPUT:

```
D:\>javac RSAAlgorithm.java

D:\>java RSAAlgorithm
Encrypted: Zz1Fr4K1jlXyn2619mWb1EwaqyoJhYfbMGF61FNzSmpx5JwnVLO7FQ9LkdVzrOFd+t6TiKYwYf29B2CrksVvtZ6VwPzEmH7ninuUUmtqiLkKT
ImQ67T1aQa28Lp9NTUYs/Ct86PZFV6XAk/GEflkLw+Z9Ziv3inG11Sfb+XBW3mUUIe2T7uesOlLdfYxXKVefA4O51BPb4hMtLbieCN8Nc7AJAcAdaA72o5VK
Z00h5JRvgFk4M1NjH6gOSbRDkzgb2fQBWFNEj3PBHlVe5TbJYADn59HO3iQPj9UZiDkmStqf41JSAWXJ++RcdBzbRqJ6fjyz+T/Cz2zERMKbxYKbA==
Decrypted: HELLO WORLD

D:\>
```