EX NO: **PLAYFAIR CIPHER USING JAVA** REG NO:210701290

DATE:

AIM:-

To implement playfair cipher substitution technique using java.

ALGORITHM:-

STEP 1: Construct a 5*5 matrix based key.

STEP 2: According to the rules,the matrix keywords are then converted to cipher text.

STEP 3: According to the table generated,split the given plain text into two's.

STEP 4: If any repeated letters exist,use filler letters.

PROGRAM:-

```
import java.util.Scanner;
public class Main {
    private static final int SIZE = 5;
    private char[][] keyMatrix = new char[SIZE][SIZE];
    private String key;
    private static final String ALPHABET =
"ABCDEFGHIKLMNOPQRSTUVWXYZ";
    public Main(String key) {
        this.key = key;
        generateKeyMatrix();
    }
```

```
private void generateKeyMatrix() {
    key = key.replaceAll("J", "I");
    key += ALPHABET;
    key = key.toUpperCase();
    key = key.replaceAll("\\s+", "");

    int index = 0;
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            keyMatrix[i][j] = key.charAt(index++);
        }
    }
}
private String preparePlainText(String plainText) {
    plainText = plainText.toUpperCase().replaceAll("\\s+", "");
    plainText = plainText.replaceAll("J", "I");
    StringBuilder preparedText = new StringBuilder(plainText);
    for (int i = 0; i < preparedText.length(); i += 2) {
        if (i + 1 == preparedText.length())
            preparedText.append('X');
        else if (preparedText.charAt(i) == preparedText.charAt(i + 1))
            preparedText.insert(i + 1, 'X');
    }
```

```
        return preparedText.toString();
    }
    private String encrypt(String plainText) {
        StringBuilder cipherText = new StringBuilder();
        for (int i = 0; i < plainText.length(); i += 2) {
            char firstChar = plainText.charAt(i);
            char secondChar = plainText.charAt(i + 1);
            int[] firstCharPosition = findPosition(firstChar);
            int[] secondCharPosition = findPosition(secondChar);
            if (firstCharPosition[0] == secondCharPosition[0]) { // Same row
                cipherText.append(keyMatrix[firstCharPosition[0]][(firstCharPosition[1]
+ 1) % SIZE]);
          cipherText.append(keyMatrix[secondCharPosition[0]][(secondCharPosition[1]
+ 1) % SIZE]);
            } else if (firstCharPosition[1] == secondCharPosition[1]) { // Same column
                cipherText.append(keyMatrix[(firstCharPosition[0] + 1) %
SIZE][firstCharPosition[1]]);
                cipherText.append(keyMatrix[(secondCharPosition[0] + 1) %
SIZE][secondCharPosition[1]]);
            } else {
cipherText.append(keyMatrix[firstCharPosition[0]][secondCharPosition[1]]);
cipherText.append(keyMatrix[secondCharPosition[0]][firstCharPosition[1]]);
            }
        }
        return cipherText.toString();
```

```
    }

    private String decrypt(String cipherText) {
        StringBuilder plainText = new StringBuilder();
        for (int i = 0; i < cipherText.length(); i += 2) {
            char firstChar = cipherText.charAt(i);
            char secondChar = cipherText.charAt(i + 1);
            int[] firstCharPosition = findPosition(firstChar);
            int[] secondCharPosition = findPosition(secondChar);
            if (firstCharPosition[0] == secondCharPosition[0]) { // Same row
                plainText.append(keyMatrix[firstCharPosition[0]][(firstCharPosition[1]
+ SIZE - 1) % SIZE]);
plainText.append(keyMatrix[secondCharPosition[0]][(secondCharPosition[1] +
SIZE - 1) % SIZE]);
            } else if (firstCharPosition[1] == secondCharPosition[1]) { // Same column
                plainText.append(keyMatrix[(firstCharPosition[0] + SIZE - 1) %
SIZE][firstCharPosition[1]]);
                plainText.append(keyMatrix[(secondCharPosition[0] + SIZE - 1) %
SIZE][secondCharPosition[1]]);
            } else { // Forming a rectangle
plainText.append(keyMatrix[firstCharPosition[0]][secondCharPosition[1]]);
plainText.append(keyMatrix[secondCharPosition[0]][firstCharPosition[1]]);
            }
        }
        return plainText.toString();
    }
```

```
private int[] findPosition(char c) {
    int[] position = new int[2];
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (keyMatrix[i][j] == c) {
                position[0] = i;
                position[1] = j;
                return position;
            }
        }
    }
    return position;
}
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter the key for Playfair Cipher:");
    String key = scanner.nextLine();
    Main playfairCipher = new Main(key);
    System.out.println("Enter the plaintext to encrypt:");
    String plainText = scanner.nextLine();
    plainText = playfairCipher.preparePlainText(plainText);
    System.out.println("Prepared plaintext: " + plainText);
```

```
        String cipherText = playfairCipher.encrypt(plainText);
        System.out.println("Encrypted ciphertext: " + cipherText);
        String decryptedText = playfairCipher.decrypt(cipherText);
        System.out.println("Decrypted plaintext: " + decryptedText);
        scanner.close();
    }
}
```

OUTPUT:-

```
Enter the key for Playfair Cipher:
5
Enter the plaintext to encrypt:
Kris
Prepared plaintext: KRIS
Encrypted ciphertext: MPHT
Decrypted plaintext: KRIS
```

RESULT:-