NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

LAB 03

LAB 03: Write a program in Java or Python to perform Cryptanalysis or decoding of Playfair Cipher

ROLL NO	52
NAME	Sarvesh Patil
CLASS	D15A
SUBJECT	Internet Security Lab
LO MAPPED	LO1: To apply the knowledge of symmetric cryptography to implement classical ciphers

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

AIM:
Write a program in Java or Python to perform Cryptanalysis or decoding of Playfair Cipher

INTRODUCTION:

What is cryptanalysis?

Cryptanalysis is the study of ciphertext, ciphers, and cryptosystems with the aim of understanding how they work and finding and improving techniques for defeating or weakening them. For example, cryptanalysts seek to decrypt ciphertexts without knowledge of the plaintext source, encryption key, or the algorithm used to encrypt it; cryptanalysts also target secure hashing, digital signatures, and other cryptographic algorithms.

How does cryptanalysis work?

While the objective of cryptanalysis is to find weaknesses in or otherwise defeat cryptographic algorithms, cryptanalysts' research results are used by cryptographers to improve and strengthen or replace flawed algorithms. Both cryptanalysis, which focuses on deciphering encrypted data, and cryptography, which focuses on creating and improving encryption ciphers and other algorithms, are aspects of cryptology, the mathematical study of codes, ciphers, and related algorithms.

Researchers may discover methods of attack that completely break an encryption algorithm, which means that ciphertext encrypted with that algorithm can be decrypted trivially without access to the encryption key. More often, cryptanalytic results uncover weaknesses in the design or implementation of the algorithm, which can reduce the number of keys that need to be tried on the target ciphertext.

For example, a cipher with a 128-bit encryption key can have 2128 (or 340,282,366,920,938,463,463,374,607,431,768,211,456) unique keys; on average, a brute force attack against that cipher will succeed only after trying half of those unique keys. If cryptanalysis of the cipher reveals an attack that can reduce the number of trials needed to 240 (or just 1,099,511,627,776) different keys, then the algorithm has been weakened significantly, to the point that a brute-force attack would be practical with commercial off-the-shelf systems.

Who uses cryptanalysis?

Cryptanalysis is practiced by a broad range of organizations, including governments aiming to decipher other nations' confidential communications; companies developing security products that employ cryptanalysts to test their security features; and hackers, crackers, independent researchers, and academicians who search for weaknesses in cryptographic protocols and algorithms.

It is this constant battle between cryptographers trying to secure information and cryptanalysts trying to break cryptosystems that moves the entire body of cryptology knowledge forward.

Cryptanalysis techniques and attacks

There are many different types of cryptanalysis attacks and techniques, which vary depending on how much information the analyst has about the ciphertext being analyzed. Some cryptanalytic methods include

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

1. In a ciphertext-only attack, the attacker only has access to one or more encrypted messages but knows nothing about the plaintext data, the encryption algorithm being used or any data about the cryptographic key being used. This is the type of challenge that intelligence agencies often face when they have intercepted encrypted communications from an opponent.
2. In a known plaintext attack, the analyst may have access to some or all of the plaintext of the ciphertext; the analyst's goal, in this case, is to discover the key used to encrypt the message and decrypt the message. Once the key is discovered, an attacker can decrypt all messages that had been encrypted using that key. Linear cryptanalysis is a type of known plaintext attack that uses a linear approximation to describe how a block cipher Known plaintext attacks depend on the attacker being able to discover or guess some or all of an encrypted message, or even the format of the original plaintext. For example, if the attacker is aware that a particular message is addressed to or about a particular person, that person's name may be a suitable known plaintext.
3. In a chosen plaintext attack, the analyst either knows the encryption algorithm or has access to the device used to do the encryption. The analyst can encrypt the chosen plaintext with the targeted algorithm to derive information about the key.
4. A differential cryptanalysis attack is a type of chosen plaintext attack on block ciphers that analyzes pairs of plaintexts rather than single plaintexts, so the analyst can determine how the targeted algorithm works when it encounters different types of data.
5. Integral cryptanalysis attacks are similar to differential cryptanalysis attacks, but instead of pairs of plaintexts, it uses sets of plaintexts in which part of the plaintext is kept constant but the rest of the plaintext is modified. This attack can be especially useful when applied to block ciphers that are based on substitution-permutation networks.
6. A side-channel attack depends on information collected from the physical system being used to encrypt or decrypt. Successful side-channel attacks use data that is neither the ciphertext resulting from the encryption process nor the plaintext to be encrypted, but rather may be related to the amount of time it takes for a system to respond to specific queries, the amount of power consumed by the encrypting system, or electromagnetic radiation emitted by the encrypting system.
7. A dictionary attack is a technique typically used against password files and exploits the human tendency to use passwords based on natural words or easily guessed sequences of letters or numbers. The dictionary attack works by encrypting all the words in a dictionary and then checking whether the resulting hash matches an encrypted password stored in the SAM file format or another password file.
8. Man-in-the-middle attacks occur when cryptanalysts find ways to insert themselves into the communication channel between two parties who wish to exchange their keys for secure communication via asymmetric or public key infrastructure The attacker then performs a key exchange with each party, with the original parties believing they are exchanging keys with each other. The two parties then end up using keys that are known to the attacker.
9. Other types of cryptanalytic attacks can include techniques for convincing individuals to reveal their passwords or encryption keys, developing Trojan horse programs that steal secret keys from victims' computers and send them back to the cryptanalyst, or tricking a victim into using a weakened cryptosystem.
10. Side-channel attacks have also been known as timing or differential power analysis. These attacks came to wide notice in the late 1990s when cryptographer Paul Kocher was

publishing results of his research into timing attacks and differential power analysis attacks on Diffie-Hellman, RSA, Digital Signature Standard (DSS), and other cryptosystems, especially against implementations on smart cards.

Tools for cryptanalysis

Because cryptanalysis is primarily a mathematical subject, the tools for doing cryptanalysis are in many cases described in academic research papers. However, there are many tools and other resources available for those interested in learning more about doing cryptanalysis. Some of them include

- CrypTool an open-source project that produces e-learning programs and a web portal for learning about cryptanalysis and cryptographic algorithms.
- Cryptol is a domain-specific language originally designed to be used by the National Security Agency specifying cryptographic algorithms. Cryptol is published under an open source license and is available for public use. Cryptol makes it possible for users to monitor how algorithms operate in software programs written to specify the algorithms or ciphers. Cryptol can be used to deal with cryptographic routines rather than with entire cryptographic suites.
- CryptoBench is a program that can be used to do cryptanalysis of ciphertext generated with many common algorithms. It can encrypt or decrypt with 29 different symmetric encryption algorithms; encrypt, decrypt, sign and verify with six different public key algorithms; and generate 14 different kinds of cryptographic hashes as well as two different types of checksum.
- Ganzúa (meaning picklock or skeleton key in Spanish) is an open-source cryptanalysis tool used for classical polyalphabetic and monoalphabetic ciphers. Ganzúa lets users define nearly completely arbitrary cipher and plain alphabets, allowing for the proper cryptanalysis of cryptograms obtained from the non-English text. A Java application, Ganzúa can run on Windows, Mac OS X, or Linux.
- Cryptanalysts commonly use many other data security tools including network sniffers and password cracking software, though it is not unusual for cryptanalytic researchers to create their own custom tools for specific tasks and challenges.

Requirements and responsibilities for cryptanalysts

A cryptanalyst's duties may include developing algorithms, ciphers, and security systems to encrypt sensitive information and data as well as analyzing and decrypting different types of hidden information, including encrypted data, cipher texts, and telecommunications protocols, in cryptographic security systems.

Government agencies as well as private sector companies hire cryptanalysts to ensure their networks are secure and sensitive data transmitted through their computer networks is encrypted.

Other duties that cryptanalysts may be responsible for include:

- Protecting critical information from being intercepted copied, modified, or deleted.
- Evaluating, analyzing, and targeting weaknesses in cryptographic security systems and algorithms.
- Designing security systems to prevent vulnerabilities.
- Developing mathematical and statistical models to analyze data and solve security problems.

- Testing computational models for accuracy and reliability.
- Investigating, researching, and testing new cryptology theories and applications.
- Searching for weaknesses in communication lines.
- Ensuring financial data is encrypted and accessible only to authorized users.
- Ensuring message transmission data isn't hacked or altered in transit.
- Decoding cryptic messages and coding systems for military, law enforcement and other government agencies.
- Developing new methods to encrypt data as well as new methods to encode messages to conceal sensitive data.

1. WHAT IS A PLAYFAIR CIPHER?

Playfair cipher is the first and best-known digraph substitution cipher, which uses the technique of symmetry encryption. Invented in 1854 by Charles Wheatstone, the cipher got its name from Lord Playfair, who promoted its use. Unlike single letters in a simple substitution cipher, the Playfair cipher technique encrypts digraphs or parts of letters.

The Playfair cipher is relatively fast and doesn't require special equipment. British Forces used it for tactical purposes during World War I and the Second Boer War, and Australians utilized it during World War II. The primary use of the cipher was for protecting vital but non-critical secrets during actual combat. By the time the enemy cryptanalysts could decrypt the information, it was useless for them.

2. PLAYFAIR CIPHER'S RELEVANCE

The Playfair cipher was significantly popular in the World War I and II era because of its complexity level compared to the then-available ciphers. Further, it didn't need any special tools or equipment to encrypt or decrypt the information.

However, after the invention of computers, the Playfair cipher was no longer used, as the computers can solve Playfair ciphers in a few seconds using break codes. Due to this reason, with the advancement of digital encryption and the passage of time, the Playfair cipher was no more an acceptable form of encoding messages as there was a risk of data getting into the wrong hands. Thus, the Playfair cipher cannot be recommended for business organizations. Further, it's a manual and labor-centric process that does not need any Playfair cipher calculator or decoder. However, there are Playfair cipher programs in C and Python.

3. PLAYFAIR CIPHER ENCRYPTION AND DECRYPTION ALGORITHM

We will have a look at some examples of Playfair Cipher with encryption and decryption algorithms.

The Playfair cipher has a key and plaintext. The key is in the form of a word which can be any sequence of 25 letters without repeats. We can build a 'key square' using this key.

Playfair Cipher Encryption Algorithm

The Playfair cipher encryption algorithm has two steps.

Generating the Key Square

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

1. The ‘key square’ is a 5×5 grid consisting of alphabets that helps encrypt the plain text.
2. All these 25 letters should be unique.
3. Since the grid can accommodate only 25 characters, there is no ‘J’ in this table. Any ‘J’ in the plaintext is replaced by ‘I’.
4. Remove any characters or punctuation that are not present in the key square. Instead, spell out the numbers, punctuations, and any other non-alphabetic text.
5. The key square will start with the key’s unique alphabet in the order of appearance, followed by the alphabet’s remaining characters in order.

Encrypting the Plain Text

Before encrypting the text, you need to divide the Playfair cipher plaintext into digraphs – pairs of two letters. In the case of plaintext with an odd number of letters, add the letter ‘Z’ to the last letter. If there are any double letters in the plain text, replace the second occurrence of the letter with ‘Z’, e.g., ‘butter’ -> ‘butzer’.

Rules for Playfair Cipher Encryption

- Case I – Both the letters in the digraph are in the same row – Consider the letters right of each alphabet. Thus, if one of the digraph letters is the rightmost alphabet in the grid, consider the leftmost alphabet in the same row.
- Case II – Both the letters in the digraph are in the same column – Consider the letters below each alphabet. Thus, if one of the digraph letters is the grid’s bottommost letter, consider the topmost alphabet in the same column.
- Case III – Neither Case I or II is true – Form a rectangle with the two letters in the digraph and consider the rectangle’s horizontal opposite corners.
- Example of Playfair Cipher Encryption Algorithm

Let’s assume that our Playfair cipher key is “Hello World,” and the plaintext that we want to encrypt is “hide the gold”.

Step 1 – Creating the key square

Our key is “Hello World”. The first characters considered from left to right and after removing the duplicates are “helowrd” as we have removed the second l in “hello” and o and l from “world”. Now let’s fill the key square with these letters. We will fill the rest of the grid with the remaining letters in the alphabet a-z without repeating any letters. Thus, our grid will look like this – *Please note that we have omitted J.

H	E	L	O	W
R	D	A	B	C
F	G	I	K	M
N	P	Q	S	T
U	V	X	Y	Z

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

Creating the Digraph

We want to encrypt the text “hide the gold”, which has an odd number of alphabets. Hence, we want to pad it with ‘Z” at the end. After splitting it into digraphs, it will look like

HI DE TH EG OL DZ

Encryption Process

There are six digraphs in total. Let’s consider the first digraph HI and locate it in the table. H and I are neither in the same row or column. Hence, applying Case III from the “Rules of Playfair Cipher Encryption” section, we have to form the rectangle using H and I and identify the horizontal opposite corners, which are L and F.

H	E	L	O	W
R	D	A	B	C
F	G	I	K	M
N	P	Q	S	T
U	V	X	Y	Z

So, the encryption for the first digraph HI -> LF

Considering the second digraph DE, the letters are in the same column. Using Case II from the “Rules of Playfair Cipher Encryption” section, consider the letters below each of them.

H	E	L	O	W
R	D	A	B	C
F	G	I	K	M
N	P	Q	S	T
U	V	X	Y	Z

So, the encryption for the digraph DE -> GD

Using these rules, the encryption for the next two digraphs is as follows –

The encryption for the digraph TH -> NW

The encryption for the digraph EG -> DP

While encrypting the next digraph OL for which the letters lie in the same row, use Case I from the “Rules of Playfair Cipher Encryption” section, and consider the letters on the right of each of them.

H	E	L	O	W
R	D	A	B	C
F	G	I	K	M

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

N	P	Q	S	T
U	V	X	Y	Z

Thus, the encryption for the digraph OL -> WO

Using these rules, the encryption for the last digraph DZ -> CV

We will summarize the digraph encryption as follows

HI -> LF

DE -> GD

TH -> NW

EG -> DP

OL -> WO

DZ -> CV

Thus, using Playfair cipher encryption rules, the encryption of “hide the gold” is “lfgdnwdpwocv”

Playfair Cipher Decryption Algorithm

The decryption of the Playfair cipher follows the same process in reverse. The receiver has the same key and key table and can decrypt the message using the key.

Rules for Playfair Cipher Decryption

Case I – Both the letters in the digraph are in the same row – Consider the letters left of each alphabet. Thus, if one of the digraph letters is the leftmost letter in the grid, consider the rightmost alphabet in the same row.
Case II – Both the letters in the digraph are in the same column – Consider the letters above each alphabet. Thus, if one of the digraph letters is the topmost letter in the grid, consider the bottommost alphabet in the same column.
Case III – Neither Case I nor II is true – Form a rectangle with the two letters in the digraph and consider the rectangle’s horizontal opposite corners.

Generating the Key Square

Generate the key square using the key texts and follow the same rules mentioned in the “Step 1 – Creating the key square” section of this example.

Creating the Digraph

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

The text that we want to decrypt is “lfgdnwdpwocv”. After splitting it into digraphs, it will look like

LF GD NW DP WO CV

Decryption Process

There are six digraphs in total. Let’s consider the first digraph LF and locate it in the table. Since L and F are neither in the same row or column, applying Case III from the “Rules of Playfair Cipher Decryption” section, we have to form the rectangle using L and F and identify the horizontal opposite corners – H and I.

H	E	L	O	W
R	D	A	B	C
F	G	I	K	M
N	P	Q	S	T
U	V	X	Y	Z

So, the decryption for the first digraph LF -> HI

In the second digraph GD, the letters are in the same column. Using Case II from the “Rules of Playfair Cipher Decryption” section, consider the letters above each alphabet.

H	E	L	O	W
R	D	A	B	C
F	G	I	K	M
N	P	Q	S	T
U	V	X	Y	Z

So, the decryption for the digraph GD -> DE

Using these rules, the decryption for the next two digraphs is as follows –

The decryption for the digraph NW -> TH

The decryption for the digraph DP -> EG

While decrypting the next digraph WO for which the letters lie in the same row, use Case I from the “Rules of Playfair Cipher Decryption” section and consider the letters on the left of each of them.

H	E	L	O	W
R	D	A	B	C
F	G	I	K	M
N	P	Q	S	T
U	V	X	Y	Z

Thus, the decryption for the digraph WO -> OL

Using these rules, the decryption for the last digraph CV -> DZ

We will summarize the digraph decryption as follows

LF -> HI

GD -> DE

NW -> TH

DP -> EG

WO -> OL

CV -> DZ

Thus, using Playfair cipher decryption rules, the decryption of “lfgdnwdpwocv” is “hide the gold”.

4. ADVANTAGES AND DISADVANTAGES OF PLAYFAIR CIPHER

With its basic features, there are specific advantages and disadvantages of Playfair cipher.

Advantages of Playfair Cipher

Relatively Difficult to Decrypt

Playfair cipher is secure and needs significant efforts to decrypt the message, making it relatively difficult to crack. At the same time, the complex mathematics behind it makes it equally difficult for the receiver to decode the information.

As the frequency analysis used for simple substitution cipher doesn’t work with the Playfair cipher, it is significantly trickier to break. Further, if one decides to undertake frequency analysis, it needs much more ciphertext on 25*25 = 625 possible digraphs than 25 possible monographs in the case of a simple substitution cipher.

Safer Alternative

The encrypting and decrypting data in the Playfair cipher is a manual method that eliminates the need for a Playfair cipher calculator. Without any need to use a Playfair cipher decoder, information can securely travel between its source and destination without getting into the wrong hands when you implement Playfair cipher encryption-decryption in C.

Disadvantages of Playfair Cipher

Symmetric Cryptography

As the Playfair cipher is a symmetric cipher, it uses the same key for encryption and decryption. One can easily crack symmetric cryptography through the Playfair cipher program technique. Also, the amount of encryption and decryption will be less.

Not Useful for Huge Data Volume

One primary disadvantage of Playfair cipher is that you can't use it to transmit massive data.

Easy to Exploit

In the Playfair cipher, the substitution is self-inverse. It means that the digraph in the ciphertext (AB) and its reverse (BA) have corresponding plaintexts, such as RU and UR. Similarly, ciphertexts UR and RU have corresponding plaintexts AB and BA). One can easily exploit it using frequency analysis, provided he/she knows the language of the plaintext.

5. EARLIER CRYPTOGRAPHIC SYSTEMS

All the earlier cryptographic systems or 'ciphers' are designed based on the symmetric key encryption scheme. They worked on alphabets as the basic elements, unlike modern digital systems that treat data as binary numbers. Following are some of the earlier cryptographic systems besides the Playfair cipher.

Caesar Cipher

Also referred to as the Shift cipher, Caesar cipher is the simplest type of substitution cipher scheme. Using this cipher, one can form a ciphertext by substituting each alphabet of plaintext by another letter, which is shifted by some fixed number lying between 0 and 25. However, it's not a secure cryptosystem as there are only 26 possible keys to decrypt the plaintext, and an attacker can easily crack it with limited computing resources.

Simple Substitution Cipher

This cipher utilizes some permutation of the letters in the alphabet. With 26 letters, the sender and receiver may select any of the 26! which is 4 x 1026 possible permutations as a ciphertext alphabet which is the secret key of the scheme. Despite the vast number of keys that the modern computing system can't comfortably break, this cipher is prone to design inefficiencies such as choosing obvious permutations. Hence, this cipher is not secured.

Vigenere Cipher

Vigenere cipher uses a text string as a key. Each alphabet in the key is converted to its numeric value, which is used to shift the alphabets in the plaintext. Since this cipher is designed by tweaking the Caesar cipher to make the cryptosystem more robust. Due to its difficulty level of decryption, it was referred to as the unbreakable cipher.

Transposition Cipher

In a transposition cipher, the order of letters in the plaintext is rearranged to form the ciphertext instead of replacing the actual plaintext letters.

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

ALGORITHM:

1. Create a matrix of 5 cross 5 is made in which all the alphabet of English letters is placed in it. Now, you must be wondering that there are 26 alphabets while the matrix is only having 25 cells. To resolve it alphabets ‘i’ and ‘j’ are placed into a single cell.
2. Now insert the key and put the remaining alphabets in the matrix. The matrix is made by inserting the value of the key and remaining alphabets into the matrix row-wise from left to right.
3. Convert the text into pairs of alphabets keeping in mind no two alphabets should repeat consecutively. For example: ‘code’ is written as ‘co’,’de’
4. If the letter is repeating then add ‘x’ to make as many pair sets as many times the alphabet is repeating. For example: ‘helloh’ is written as ‘he’ ‘lx‘, ‘lo‘, ‘hx’.
5. Solve the matrix or form code using 3 standard rules
6. If both the alphabet are in the same row, replace them with alphabets to their immediate right.
7. If both the alphabets are in the same column, replace them with alphabets immediately below them.
8. If not in the same row or column, replace them with alphabets in the same row respectively, but at other pair of corners

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

PROGRAM:

```
import java.awt.Point;
import java.util.Scanner;

public class Main {
   //length of digraph array
   private int length = 0;
   //creates a matrix for Playfair cipher
   private String[][] table;

   //main() method to test Playfair method
   public static void main(String args[]) {
      Main pf = new Main();
   }

   //main run of the program, Playfair method
//constructor of the class
   private Main() {
//prompts user for the keyword to use for encoding & creates tables
      System.out.print("Enter the key for playfair cipher: ");
      Scanner sc = new Scanner(System.in);
      String key = parseString(sc);
      while (key.equals(""))
         key = parseString(sc);
      table = this.cipherTable(key);
//prompts user for message to be encoded
      System.out.print("Enter the plaintext to be encipher: ");
//System.out.println("using the previously given keyword");
      String input = parseString(sc);
      while (input.equals(""))
         input = parseString(sc);
//encodes and then decodes the encoded message
      String output = cipher(input);
      String decodedOutput = decode(output);
//output the results to user
      this.keyTable(table);
      this.printResults(output, decodedOutput);
   }

   //parses an input string to remove numbers, punctuation,
//replaces any J's with I's and makes string all caps
   private String parseString(Scanner sc) {
      String parse = sc.nextLine();
//converts all the letters in upper case
      parse = parse.toUpperCase();
```

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

```
//the string to be substituted by space for each match (A to Z)
      parse = parse.replaceAll("[^A-Z]", "");
//replace the letter J by I
      parse = parse.replace("J", "I");
      return parse;
   }

   //creates the cipher table based on some input string (already parsed)
   private String[][] cipherTable(String key) {
//creates a matrix of 5*5
      String[][] playfairTable = new String[5][5];
      String keyString = key + "ABCDEFGHIKLMNOPQRSTUVWXYZ";
//fill string array with empty string
      for (int i = 0; i < 5; i++)
         for (int j = 0; j < 5; j++)
            playfairTable[i][j] = "";
      for (int k = 0; k < keyString.length(); k++) {
         boolean repeat = false;
         boolean used = false;
         for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++) {
               if (playfairTable[i][j].equals("" + keyString.charAt(k))) {
                  repeat = true;
               } else if (playfairTable[i][j].equals("") && !repeat && !used) {
                  playfairTable[i][j] = "" + keyString.charAt(k);
                  used = true;
               }
            }
         }
      }
      return playfairTable;
   }

   //cipher: takes input (all upper-case), encodes it, and returns the output
   private String cipher(String in) {
      length = (int) in.length() / 2 + in.length() % 2;
//insert x between double-letter digraphs & redefines "length"

      for (int i = 0; i < (length - 1); i++) {
         if (in.charAt(2 * i) == in.charAt(2 * i + 1)) {
            in = new StringBuffer(in).insert(2 * i + 1, 'X').toString();
            length = (int) in.length() / 2 + in.length() % 2;
         }
      }
//----------makes plaintext of even length----------
//creates an array of digraphs
```

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

```
        String[] digraph = new String[length];
//loop iterates over the plaintext
        for (int j = 0; j < length; j++) {
//checks the plaintext is of even length or not
            if (j == (length - 1) && in.length() / 2 == (length - 1))
//if not addends X at the end of the plaintext
                in = in + "X";
            digraph[j] = in.charAt(2 * j) + "" + in.charAt(2 * j + 1);
        }
        System.out.println();
        System.out.println("Digraph");
        for (int l = 0; l< length; l++) {
            System.out.println(digraph[l]);
        }
//encodes the digraphs and returns the output
        String out = "";
        String[] encDigraphs = new String[length];
        encDigraphs = encodeDigraph(digraph);
        for (int k = 0; k < length; k++)
            out = out + encDigraphs[k];
        return out;
    }

    //--------------encryption logic----------------
//encodes the digraph input with the cipher's specifications
    private String[] encodeDigraph(String di[]) {
        String[] encipher = new String[length];
        for (int i = 0; i < length; i++) {
            char a = di[i].charAt(0);
            char b = di[i].charAt(1);
            int r1 = (int) getPoint(a).getX();
            int r2 = (int) getPoint(b).getX();
            int c1 = (int) getPoint(a).getY();
            int c2 = (int) getPoint(b).getY();
//executes if the letters of digraph appear in the same row
//in such case shift columns to right
            if (r1 == r2) {
                c1 = (c1 + 1) % 5;
                c2 = (c2 + 1) % 5;
            }
//executes if the letters of digraph appear in the same column
//in such case shift rows down
            else if (c1 == c2) {
                r1 = (r1 + 1) % 5;
                r2 = (r2 + 1) % 5;
            }
```

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

```
//executes if the letters of digraph appear in the different row and different column
//in such case swap the first column with the second column
        else {
           int temp = c1;
           c1 = c2;
           c2 = temp;
        }
//performs the table look-up and puts those values into the encoded array
        encipher[i] = table[r1][c1] + "" + table[r2][c2];
     }
     return encipher;
  }

  //----------------------decryption logic----------------------
// decodes the output given from the cipher and decode methods (opp. of encoding process)
  private String decode(String out) {
     String decoded = "";
     for (int i = 0; i < out.length() / 2; i++) {
        char a = out.charAt(2 * i);
        char b = out.charAt(2 * i + 1);
        int r1 = (int) getPoint(a).getX();
        int r2 = (int) getPoint(b).getX();
        int c1 = (int) getPoint(a).getY();
        int c2 = (int) getPoint(b).getY();
        if (r1 == r2) {
           c1 = (c1 + 4) % 5;
           c2 = (c2 + 4) % 5;
        } else if (c1 == c2) {
           r1 = (r1 + 4) % 5;
           r2 = (r2 + 4) % 5;
        } else {
//swapping logic
           int temp = c1;
           c1 = c2;
           c2 = temp;
        }
        decoded = decoded + table[r1][c1] + table[r2][c2];
     }
//returns the decoded message
     return decoded;
  }

  // returns a point containing the row and column of the letter
  private Point getPoint(char c) {
     Point pt = new Point(0, 0);
     for (int i = 0; i < 5; i++)
```

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

```
            for (int j = 0; j < 5; j++)
                if (c == table[i][j].charAt(0))
                    pt = new Point(i, j);
        return pt;
    }

    //function prints the key-table in matrix form for playfair cipher
    private void keyTable(String[][] printTable) {
        System.out.println();
        System.out.println("Playfair Cipher Key Matrix: ");
//loop iterates for rows
        for (int i = 0; i < 5; i++) {
//loop iterates for column
            for (int j = 0; j < 5; j++) {
//prints the key-table in matrix form
                System.out.print(printTable[i][j] + " ");
            }
            System.out.println();
        }
        System.out.println();
    }

    //method that prints all the results
    private void printResults(String encipher, String dec) {
        System.out.print("Encrypted Message: ");
//prints the encrypted message
        System.out.println(encipher);
        System.out.println();
        System.out.print("Decrypted Message: ");
//prints the decryted message
        System.out.println(dec);
    }
}
```

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

RESULTS:

TEST CASE 1:

```
/Library/Java/JavaVirtualMachines/jdk-18.0.2.jdk/Contents/Home/bin/java
Enter the key for playfair cipher: sarvesh
Enter the plaintext to be encipher: My name is Sarvesh Patil

Digraph
MY
NA
ME
IS
SA
RV
ES
HP
AT
IL

Playfair Cipher Key Matrix:
S A R V E
H B C D F
G I K L M
N O P Q T
U W X Y Z

Encrypted Message: LZOSTFGAARVESACNEOKM

Decrypted Message: MYNAMEISSARVESHPATIL
```

TEST CASE 2:

```
/Library/Java/JavaVirtualMachines/jdk-18.0.2.jdk/Contents/Home/bin/java
Enter the key for playfair cipher: sarvesh
Enter the plaintext to be encipher: Hello my name is Sarvesh

Digraph
HE
LX
LO
MY
NA
ME
IS
SA
RV
ES
HX

Playfair Cipher Key Matrix:
S A R V E
H B C D F
G I K L M
N O P Q T
U W X Y Z

Encrypted Message: FSKYIQLZOSTFGAARVESACU

Decrypted Message: HELXLOMYNAMEISSARVESHX
```

CONCLUSION:

We have successfully understood the implementation of a playfair cipher.