Practical No. 2

Aim:- To write C/Java program to implement playfair cipher.

Theory:- The playfair cipher was the first practical diagraph substitution cipher. It was invented in 1884 by Cardef wheatstone but was named after lord playfair who promoted the used of the cipher. In playfair cipher unlike traditional cipher we encrypt a pair of alphabets instead of single alphabets.

It was used for tactical purpose by British forces in the second Boer war and in world war. It was used for the same purpose by the australian during world war II.

The playfair cipher encryption algorithm:-

1> Generate the key square (5×5)

* The key square is a 5×5 grid of alphabets that acts as key for encrypting the plaintext. Each of the 25 alphabets must be unique and one letter of the alphabets is omitted from the table. If alphabet contains J, then it is replaced by J.

* The initial alphabets in the key squares are the unique alphabets of the key order in which the appeared follow by the remaining letters of the alphabets in order.

2. Algorithm to encrypt the plaintext:-

The plaintext is split into pairs of two letters (diagraph). If there is an odd numbers of letters, a Z is added to the last letter.

What is playfair cipher?.

The playfair cipher encryption technique can be used to encrypt or encode a message. It operates exactly like typical encryption. The only difference is that it encryption a diagraph or a pair of two letters instead of single letters.

An initial 5x5 matrix key table is created. The plaintext encryption key is made out of matrix alphabet characters. Be mindful that you shouldnt repeat the letters. There are 26 alphabets. However, there are only 25 spaces in which we can place a letter. The matrix will delete the extra letter because there is an excess of one letter.

This blog provides a full explaination of play-fair cipher, its advantages and disadvantages its applicability and the playfair encryption and decryption algorithm.

The playfair cipher is constrained by the following:—

- only 25 alphabets are supported.
- It is incompatible with characters other that numbers.

Conclusion:— Hence, we have successfully implemented playfair cipher.

Ashwaka2

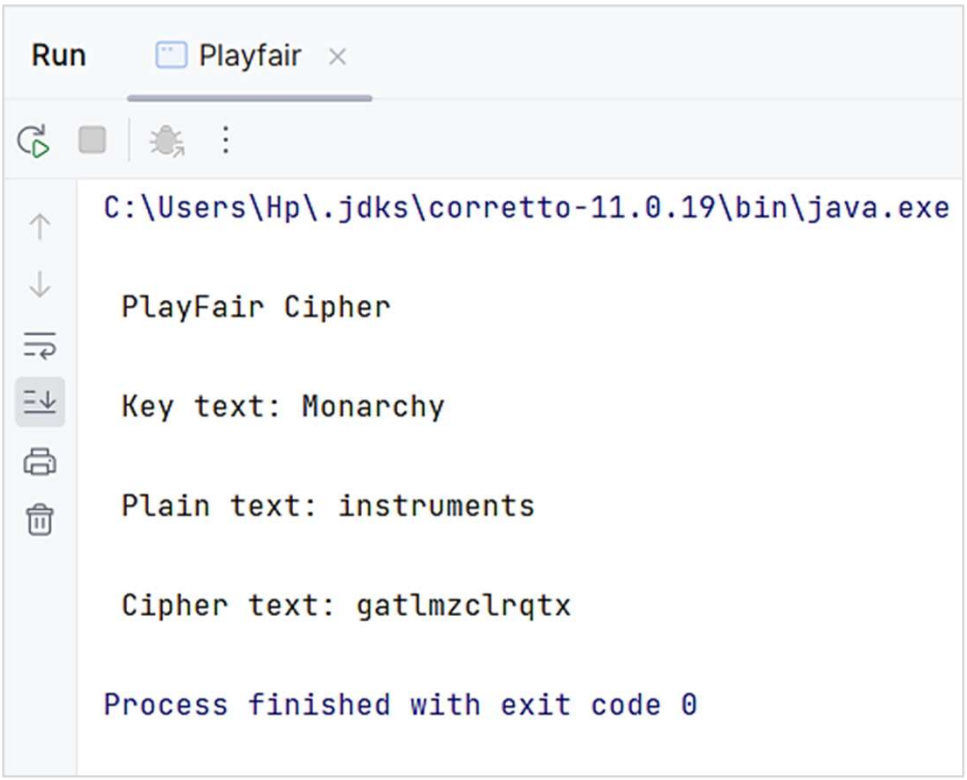
Program:

```
import java.util.*;
public class Solution {
   static int SIZE = 30;

   // Function to convert the string to lowercase
   static void toLowerCase(char plain[], int ps) {
      int i;
      for (i = 0; i < ps; i++)
         if (plain[i] > 64 && plain[i] < 91)
            plain[i] += 32;
   }

   // Function to remove all spaces in a string
   static int removeSpaces(char[] plain, int ps) {
      int i, count = 0;
      for (i = 0; i < ps; i++)
         if (plain[i] != '\u0000')
            plain[count++] = plain[i];
      return count;
   }

   // Function to generate the 5x5 key square
   static void generateKeyTable(char key[], int ks, char keyT[][]) {
      int i, j, k, flag = 0;
      // a 26 character hashmap to store count of the alphabet
      int dicty[] = new int[26];
      for (i = 0; i < ks; i++) {
         if (key[i] != 'j')
            dicty[key[i] - 97] = 2;
      }
      dicty['j' - 97] = 1;
      i = 0;
      j = 0;
      for (k = 0; k < ks; k++) {
         if (dicty[key[k] - 97] == 2) {
            dicty[key[k] - 97] -= 1;
            keyT[i][j] = key[k];
            j++;
            if (j == 5) {
               i++;
               j = 0;
            }
         }
      }
      for (k = 0; k < 26; k++) {
         if (dicty[k] == 0) {
            keyT[i][j] = (char)(k + 97);
            j++;
            if (j == 5) {
               i++;
               j = 0;
            }
         }
      }
   }
```

```
// Function to search for the characters of a digraph
// in the key square and return their position
static void search(char keyT[][], char a, char b, int arr[]) {
   int i, j;
   if (a == 'j')
      a = 'i';
   else if (b == 'j')
      b = 'i';
   for (i = 0; i < 5; i++) {
      for (j = 0; j < 5; j++) {
         if (keyT[i][j] == a) {
            arr[0] = i;
            arr[1] = j;
         } else if (keyT[i][j] == b) {
            arr[2] = i;
            arr[3] = j;
         }
      }
   }
}

// Function to find the modulus with 5
static int mod5(int a) {
   return (a % 5);
}

// Function to make the plain text length to be even
static int prepare(char str[], int ptrs) {
   if (ptrs % 2 != 0) {
      str[ptrs++] = 'z';
      str[ptrs] = '\0';
   }
   return ptrs;
}

// Function for performing the encryption
static void encrypt(char str[], char keyT[][], int ps) {
   int i;
   int[] a =new int[4];
   for (i = 0; i < ps; i += 2) {
      search(keyT, str[i], str[i + 1], a);
      if (a[0] == a[2]) {
         str[i] = keyT[a[0]][mod5(a[1] + 1)];
         str[i + 1] = keyT[a[0]][mod5(a[3] + 1)];
      } else if (a[1] == a[3]) {
         str[i] = keyT[mod5(a[0] + 1)][a[1]];
         str[i + 1] = keyT[mod5(a[2] + 1)][a[1]];
      } else {
         str[i] = keyT[a[0]][a[3]];
         str[i + 1] = keyT[a[2]][a[1]];
      }
   }
}
```

```
// Function to encrypt using Playfair Cipher
   static void encryptByPlayfairCipher(char str[], char key[]) {
      int ps;
      int ks;
      char[][] keyT = new char[5][5];

      // Key
      ks = key.length;
      ks = removeSpaces(key, ks);
      toLowerCase(key, ks);

      // Plaintext
      ps = str.length;
      toLowerCase(str, ps);
      ps = removeSpaces(str, ps);
      ps = prepare(str, ps);
      generateKeyTable(key, ks, keyT);
      encrypt(str, keyT, ps);
   }

   static void strcpy(char[] arr, String s) {
      for(int i = 0; i < s.length(); i++) {
         arr[i] = s.charAt(i);
      }
   }

// Driver code
   public static void main(String[] args) {
      char str[] = new char[SIZE];
      char key[] = new char[SIZE];
      System.out.println(“PlayFair Cipher”);
      // Key to be encrypted
      strcpy(key, "Monarchy");
      System.out.println("Key text: " + String.valueOf(key));

      // Plaintext to be encrypted
      strcpy(str, "instruments");
      System.out.println("Plain text: " + String.valueOf(str));

      // encrypt using Playfair Cipher
      encryptByPlayfairCipher(str, key);
      System.out.println("Cipher text: " + String.valueOf(str));
   }
}
```

Output:


Run
Playfair
C:\Users\Hp\.jdks\corretto-11.0.19\bin\java.exe
PlayFair Cipher
Key text: Monarchy
Plain text: instruments
Cipher text: gatlmzclrqtx
Process finished with exit code 0