Lab1

Shlok Kamath

21BAI1844

Cryptography and Network Security Lab (BCSE309P)

Professor: Dr. RAJESH R

Today's task:

1. CEASAR CIPHER
2. PLAYFAIR CIPHER

Ceasar Cipher:

$En(x)=(x+n) \bmod 26$
(Encryption Phase with shift n)

$Dn(x)=(x-n) \bmod 26$
(Decryption Phase with shift n)

Code:

```
import java.util.*;

class Main {
    public static void main(String[] args) {
        char[] arr = new char[26];
        int j = 0;
        for (char i = 'a' ; i <= 'z' ; i++)
            arr[j++] = i;
        Scanner sc = new Scanner(System.in);
        String s = sc.next();
        String c = sc.next();
        int i=0;
        while(c.length()!=s.length())
            c+=c.charAt(i++);
        String en = "";
        for(i=0; i<c.length(); i++)
            en+=arr[(s.charAt(i) + c.charAt(i) - 'a' - 'a')%26];
        System.out.println(en);
        String decoded = "";
        for(i=0; i<c.length(); i++)
            decoded+=arr[(en.charAt(i) - c.charAt(i))%26];
        System.out.println(decoded);

    }
}
```

Output:

```
Run:  Main
C:\Users\kamat\.jdks\openjdk-17.0.2\bin\java.exe
jaishriram
abcdefghij
jbkvlwoyiv
jaishriram

Process finished with exit code 0
```

PLAYFAIR CIPHER:

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

    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]];
            }
        }
    }

    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);
        }
    }
```

```java
    // Driver code
    public static void main(String[] args) {
        char str[] = new char[SIZE];
        char key[] = new char[SIZE];

        // Key to be encrypted

        strcpy(key, "shlok");
        System.out.println("Key text: " + String.valueOf(key).trim());

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

        // encrypt using Playfair Cipher
        encryptByPlayfairCipher(str, key);

        System.out.println("Cipher text: " + String.valueOf(str).trim());
    }

}
```

Output:

```
Run: Main
C:\Users\kamat\.jdks\openjdk-17.0.2\bin\java.exe
Key text: shlok
Plain text: jaishriram
Cipher text: fcfllqrxdf

Process finished with exit code 0
```