**Ministerul Educaţiei Tineretului şi Sportului al Republicii Moldova**

**Universitatea Tehnică a Moldovei**

**RAPORT**

**LA CRIPTOGRAFIE**

Tema: Cifruri polialfabetice

Lucrarea de Laborator nr. 3

A efectuat Studentul grupei \_SI-212\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_Vozian Vladimir\_\_\_\_

*semnătura nume, prenume*

A verificat \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*nume, prenume profesor*

**Chisinău 2022**

## Sarcina 1

## Algoritmul Vigenere

## Sarcină 3.2. De implementat algoritmul Vigenere în unul din limbajele de programare pentru mesaje în limba română (31 de litere), acestea fiind codificate cu numerele 0, 1, ... 30. Valorile caracterelor textului sunt cuprinse între ‘A’ și ’Z’, ’a’ și ’z’ și nu sunt premise alte valori. În cazul în care utilizatorul introduce alte valori - i se va sugera diapazonul corect al caracterelor. Lungimea cheii nu trebuie să fie mai mică de 7. Criptarea și decriptarea se va realiza în conformitate cu formulele din modelul matematic prezentat mai sus. În mesaj mai întâi trebuie eliminate spațiile, apoi toate literele se vor transforma în majuscule. Utilizatorul va putea alege operația - *criptare* sau *decriptare*, va putea introduce *cheia*, *mesajul* sau *criptograma* și va obține *criptograma* sau *mesajul decriptat*.

## Mersul lucrării

Criptarea

Codul:

import numpy as np

alphabet = "AĂÂBCDEFGHIÎKLMNOPQRSȘTȚUVWXYZ"

def Criptare():

    message = input("Mesajul => ").upper()

    message = message.replace(" ","")

    key = input("Cheia => ").upper()

    key = key.replace(" ","")

    cripted = ""

    key1=key

    while (len(key)<=len(message)):

        key+=key1

    key = key[0: len(message)]

    for i in range(0,len(message)):

        position\_m = alphabet.find(message[i])

        position\_k = alphabet.find(key[i])

        position\_c = (position\_m + position\_k) % 31

        cripted += alphabet[position\_c]

    print(cripted)

def Decriptare():

    cripted = input("Mesajul criptat => ").upper()

    cripted = cripted.replace(" ","")

    key = input("Cheia => ").upper()

    key = key.replace(" ","")

    message = ""

    key1=key

    while (len(key)<=len(cripted)):

        key+=key1

    key = key[0: len(cripted)]

#    print(cripted)

#    print(key)

    for i in range(0,len(cripted)):

        position\_c = alphabet.find(cripted[i])

        position\_k = alphabet.find(key[i])

        position\_m = (position\_c - position\_k) % 31

        message += alphabet[position\_m]

    print(message)

menu=input("""

1-Criptare

2-Decriptare

0-Exit

""")

match menu:

    case "1":

        Criptare()

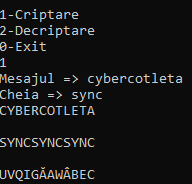
    case "2":

        Decriptare()

Rezultat:

Pentru criptare ca date de intrare sa luat mesajul “cybercotleta” si cheia „sync”, unde literele sunt mici, in codul programului este algoritmul care automat face toate literele mari.

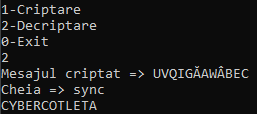
La fel se vede ca cheia se repeta corect ca sa fie de lungimea mesajului.



Decriptarea

Rezultat:

Pentru decriptare sa luat mesajul codificat primit de la criptare, care necesită deja decriptarea



## Sarcina 2

## Algoritmul Playfair

## Sarcină 3.1. De implementat algoritmul Playfair în unul din limbajele de programare pentru mesaje în limba română (31 de litere). Valorile caracterelor textului sunt cuprinse între ‘A’ și ’Z’, ’a’ și ’z’ și nu sunt premise alte valori. În cazul în care utilizatorul introduce alte valori - i se va sugera diapazonul corect al caracterelor. Lungimea cheii nu trebuie să fie mai mică de 7. Utilizatorul va putea alege operația - *criptare* sau *decriptare*, va putea introduce *cheia*, *mesajul* sau *criptograma* și va obține *criptograma* sau *mesajul decriptat*. Faza finală de adăugare a spațiilor noi, în funcție de limba folosită și de logica mesajului – se va face manual.

Codul:

import numpy as np

import random

alphabet='AĂÂBCDEFGHIÎKLMNOPQRSȘTȚUVWXYZ'

inlocuire ='QVXZ'

menu=int(input("""

1-Criptare

2-Decriptare

0-Exit

"""))

def Criptare():

    message = input("Mesajul => ").upper()

    message = message.replace(" ","")

    def verificare():

        for i in range(0, len(message)):

            if message[i] not in alphabet:

                return False

        return True

    while verificare() == False:

        print("Unele caractere introduse nu sunt din alfabetul roman!")

        print("Introduceti inca o data dar in diapazonul alfabetului roman: AĂÂBCDEFGHIÎKLMNOPQRSȘTȚUVWXYZ")

        message = input("Mesaj =>").upper()

        verificare()

    def replacedChar():

        prev = ''

        res = ''

        res = inlocuire[random.randint(0, len(inlocuire) - 1)]

        while res == prev:

            res = inlocuire[random.randint(0, len(inlocuire) - 1)]

        prev = res

        return res

    for i in range(len(message)):

        if(message[i] == message[i - 1] and message[i+1] == message[i]):

            message = message[:i] + replacedChar() + message[i:]

            message = message[:i+2] + replacedChar() + message[i+2:]

    message = " ".join(message[i:i+2] for i in range(0, len(message), 2))

    for i in range(len(message)):

        if(message[i] == message[i - 1]):

            message = message[:i] + replacedChar() + message[i:]

            message = message[:i+1] + ' ' + message[i+1:]

    message = message.replace(" ","")

    if (len(message) % 2 != 0):

        message = message[0: len(message)-1]

    message = " ".join(message[i:i+2] for i in range(0, len(message), 2))

    print(message)

    key = input("key => ").upper()

    key = key.replace(" ","")

    matrix\_6x5 = [[0 for i in range (6)] for j in range(5)]

    need\_alph = ''.join(dict.fromkeys(key + alphabet))

    index = 0

    for i in range(0,5):

        for j in range(0,6):

            matrix\_6x5[i][j] = need\_alph[index]

            index+=1

    for i in matrix\_6x5:

        print(i)

    def indexLocator (char):

        indexOfChar = []

        for i,j in enumerate(matrix\_6x5):

            for k,l in enumerate(j):

                if char == l:

                    indexOfChar.append(i)

                    indexOfChar.append(k)

                    return indexOfChar

    message = message.replace(" ","")

    cipherText = []

    i = 0

    while i < len(message):

        n1 = indexLocator(message[i])

        n2 = indexLocator(message[i+1])

        if (n1[1] == n2[1]):

            i1 = (n1[0] + 1) % 5

            j1 = n1[1]

            i2 = (n2[0] + 1) % 5

            j2 = n2[1]

            cipherText.append(matrix\_6x5[i1][j1])

            cipherText.append(matrix\_6x5[i2][j2])

        elif (n1[0]==n2[0]):

            i1= n1[0]

            j1= (n1[1] + 1) % 6

            i2= n2[0]

            j2= (n2[1] + 1) % 6

            cipherText.append(matrix\_6x5[i1][j1])

            cipherText.append(matrix\_6x5[i2][j2])

        else:

            i1 = n1[0]

            j1 = n1[1]

            i2 = n2[0]

            j2 = n2[1]

            cipherText.append(matrix\_6x5[i1][j2])

            cipherText.append(matrix\_6x5[i2][j1])

        i += 2

    def listToString():

        str1 = ""

        for ele in cipherText:

            str1 += ele

        str1 = " ".join(str1[i:i+2] for i in range(0, len(str1), 2))

        return str1

    print(listToString())

def Decriptare():

    message = input("Textul criptat => ").upper()

    #message = "CONGRESS SHALL MAKE NO LAW".upper()

    message = message.replace(" ","")

    message = " ".join(message[i:i+2] for i in range(0, len(message), 2))

    key = input("key => ").upper()

    key = key.replace(" ","")

    while len(key)<7:

        print("Lungimea cheii este pea mica, necesita sa fie lungimea nu mai mica de 7")

        key = input("Key =>").upper()

    matrix\_6x5 = [[0 for i in range (6)] for j in range(5)]

    need\_alph = ''.join(dict.fromkeys(key + alphabet))

    index = 0

    for i in range(0,5):

        for j in range(0,6):

            matrix\_6x5[i][j] = need\_alph[index]

            index+=1

    for i in matrix\_6x5:

        print(i)

    def indexLocator (char):

        indexOfChar = []

        for i,j in enumerate(matrix\_6x5):

            for k,l in enumerate(j):

                if char == l:

                    indexOfChar.append(i)

                    indexOfChar.append(k)

                    return indexOfChar

    message = message.replace(" ","")

    cipherText = []

    i = 0

    while i < len(message):

        n1 = indexLocator(message[i])

        n2 = indexLocator(message[i+1])

        if (n1[1] == n2[1]):

            i1 = (n1[0] - 1) % 5

            j1 = n1[1]

            i2 = (n2[0] - 1) % 5

            j2 = n2[1]

            cipherText.append(matrix\_6x5[i1][j1])

            cipherText.append(matrix\_6x5[i2][j2])

        elif (n1[0]==n2[0]):

            i1= n1[0]

            j1= (n1[1] - 1) % 6

            i2 = n2[0]

            j2 = (n2[1] - 1) % 6

            cipherText.append(matrix\_6x5[i1][j1])

            cipherText.append(matrix\_6x5[i2][j2])

        else:

            i1 = n1[0]

            j1 = n1[1]

            i2 = n2[0]

            j2 = n2[1]

            cipherText.append(matrix\_6x5[i1][j2])

            cipherText.append(matrix\_6x5[i2][j1])

        i += 2

    def listToString():

        str1 = ""

        for ele in cipherText:

            str1 += ele

        str1 = str1.replace("Q","")

        str1 = str1.replace("V","")

        str1 = str1.replace("X","")

        str1 = str1.replace("Z","")

        return str1

    print(listToString())

match menu:

    case 1:

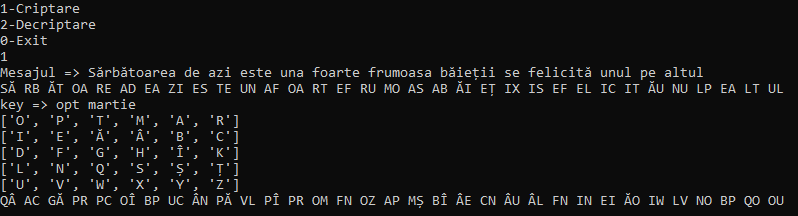
        Criptare()

    case 2:

        Decriptare()

Rezultat:

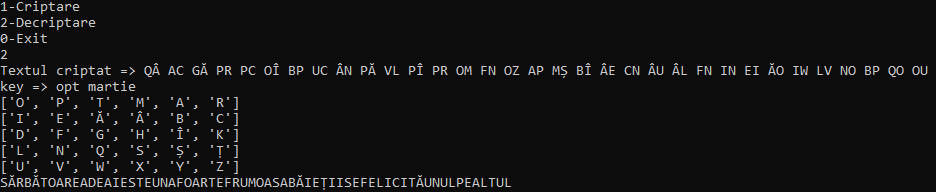
Pentru criptare sa luat ca set de date mesajul „Sărbătoarea de azi este una foarte frumoasa băieții se felicită unul pe altul”, si cu cheia „opt martie”.



Decriptare

Rezultat:

Pentru decriptare iarăși luăm rezultatul primit la criptare cu aceeași cheie și cuvânt pentru criptare.



Concluzii:

Lucrarea dată era efectuată in python, lucrul efectuat cu primul cifru Vigenere nu a fost dificil, sarcina nu este asa ampla si cifrul este mai compact, dar cu cifrul Playfair au fost intalnite unele dificultati, indeosebi in lucrul cu coordonatele caracterelor in matrice si schimbul datorita lor a mesajului.