Tugas 3

Praktikum Kriptografi

Nama : M Nabil Fikri S P

NPM : 140810200046

Penjelasan Program:

```
import numpy as np
import pymatrix


def main():


    choice = ""

    while True:
        print("================================")
        print("HILL CIPHER")
        print("1. Encrypt")
        print("2. Decrypt")
        print("3. Find Key")
        print("4. Exit")
        choice = input("Please chose action (1/2/3): ")

        match choice:
            case "1":
                plainText = input("Input plaintext: ")
                size = int()
                loop = True
                while loop:
                    size = int(input("input key matrix size (2/3): "))
                    if size == 2:
                        loop = False

                    elif size == 3:
                        loop = False

                    else:
                        print("matrix size can only be 2x2 or 3x3
(2/3)!")
```

```
                key = createKey(size)
                cipherText = encryption(key, plainText)
                print("Encrypted text: " + cipherText)
            case "2":
                cipherText = input("Input ciphertext: ")
                size = int()
                loop = True
                while loop:
                    size = int(input("input key matrix size (2/3): "))
                    if size == 2:
                        loop = False

                    elif size == 3:
                        loop = False

                    else:
                        print("matrix size can only be 2x2 or 3x3
(2/3)!")

                key = createKey(size)
                plainText = decryption(key, cipherText)
                print("Encrypted text: " + plainText)
            case "3":
                plainText = input("input plaintext: ")
                cipherText = input("input ciphertext: ")
                size = int()
                loop = True
                while loop:
                    size = int(input("input key matrix size (2/3): "))
                    if size == 2:
                        loop = False

                    elif size == 3:
                        loop = False

                    else:
                        print("matrix size can only be 2x2 or 3x3
(2/3)!")
                print(findKey(plainText, cipherText, size))
            case "4":
                print("BYE BYE")
                break
            case _:
                print("please choose given choice")
```

```
def createKey(size) -> list:
    key = np.empty([size, size], dtype=int)
    for i in range(size):
        for j in range(size):
            key[i][j] = input("input " + str(i) + ", " + str(j) + "
element: ")
    return key


def encryption(key: list, text: str) -> str:

    # catat index non alphabet supaya dihapus dan tidak dienkripsi
    symbol = {}
    text_list = list(text)
    for i in range(len(text)):
        if not text[i].isalpha():
            symbol[i] = text[i]
            text_list.remove(text[i])
    text = ''.join(text_list)

    # menambah huruf x jika panjang text tidak kelipatan dengan besar
matrix
    initLenText = len(text)
    while len(text) % len(key) != 0:
        text += 'x'

    # condition handling untuk uppercase
    upper = []
    for i in range(len(text)):
        if text[i].isupper():
            upper.append(i)

    text = text.lower()

    textMCol = int(len(text) / len(key))
    textMRow = int(len(key))
    textMatrix = np.empty([textMRow, textMCol], dtype=int)

    textIndex = 0
    for i in range(textMCol):
        for j in range(textMRow):
            textMatrix[j][i] = ord(text[textIndex + j]) - ord('a')
        textIndex += len(key)
```

```
    encrypted = np.matmul(key, textMatrix)

    for i in range(len(encrypted)):
        for j in range(len(encrypted[0])):
            encrypted[i][j] = encrypted[i][j] % 26 + ord('a')

    result = str()
    temp = str()
    for i in range(len(encrypted[0])):
        temp = ""
        for j in range(len(encrypted)):
            temp += chr(encrypted[j][i])
        result += temp

    # penghapusan huruf jika kalimat ditambah 'x'
    # if initLenText != len(result):
    #     result = result[:-(len(result)-initLenText)]

    for i in range(len(upper)):
        result = result[:upper[i]] + \
            result[upper[i]].swapcase() + result[upper[i]+1:]

    for x, y in symbol.items():
        result = result[:x] + y + result[x:]

    return result


def decryption(key: list, text: str) -> str:
    inversed = inverseMatrix(key)

    if (gcd(determinant(inversed) % 26, 26)) != 1:
        print("matrix tidak inversible 1")
        return ""
    else:
        return encryption(inversed, text)


def gcd(a: int, b: int) -> int:
    while b != 0:
        temp = int(a)
        a = b
        b = temp % b
    return a
```

```
def modInverse(a: int, b: int) -> int:
    for i in range(1, b):
        if (((a % b) * (i % b)) % b == 1):
            return i
    return -1


def determinant(key) -> int:
    return int((np.linalg.det(key)).round())


def getAdjoint(key: list) -> list:
    result = np.copy(key)
    if len(key) == 2:
        temp = int()
        result[0][1] = -1 * result[0][1]
        result[1][0] = -1 * result[1][0]

        temp = result[0][0]
        result[0][0] = result[1][1]
        result[1][1] = temp
        return result

    elif len(key) == 3:
        result[0][0] = (key[1][1]*key[2][2] - key[2][1]*key[1][2])
        result[0][1] = ((key[1][0]*key[2][2] - key[1][2]*key[2][0]) * -
1)
        result[0][2] = (key[1][0]*key[2][1] - key[1][1]*key[2][0])
        result[1][0] = ((key[0][1]*key[2][2] - key[0][2]*key[2][1]) * -
1)
        result[1][1] = (key[0][0]*key[2][2] - key[0][2]*key[2][0])
        result[1][2] = ((key[0][0]*key[2][1] - key[0][1]*key[2][0]) * -
1)
        result[2][0] = (key[0][1]*key[1][2] - key[0][2]*key[1][1])
        result[2][1] = ((key[0][0]*key[1][2] - key[0][2]*key[1][0]) * -
1)
        result[2][2] = ((key[0][0]*key[1][1] - key[0][1]*key[1][0]))
        result = result.transpose()
        return result

    else:
        cof = pymatrix.Matrix.from_list(key.tolist())
        cof = cof.adjoint()
```

```
        for i in range(len(key)):
            for j in range(len(key[0])):
                result[i][j] = cof[i][j]
        return result


def inverseMatrix(key: list) -> list:
    inversed = np.copy(key)
    D = modInverse(determinant(key), 26)

    if gcd(D, 26) != 1:
        print("matrix tidak inversible 2")
        return

    adj = getAdjoint(key)
    for i in range(len(adj)):
        for j in range(len(adj[0])):
            adj[i][j] = adj[i][j] % 26

    for i in range(len(key)):
        for j in range(len(key[0])):
            inversed[i][j] = (adj[i][j] * D) % 26
    return inversed


def findKey(plainText: str, cipherText: str, keySize: int) -> list:
    matrixP = np.empty([keySize, keySize], dtype=int)
    matrixC = np.empty([keySize, keySize], dtype=int)
    key = np.empty([keySize, keySize], dtype=int)

    textIndex = int(0)
    for i in range(keySize):
        for j in range(keySize):
            matrixP[j][i] = ord(plainText[textIndex + j]) - ord('a')
            matrixC[j][i] = ord(cipherText[textIndex + j]) - ord('a')
        textIndex += keySize

    if (gcd(modInverse(determinant(matrixP) % 26, 26) % 26, 26)) != 1:
        print("matrix tidak inversible 3")
        return

    matrixP = inverseMatrix(matrixP)

    key = np.matmul(matrixC, matrixP)
```

```
    # for i in range(keySize):
    #     for j in range(keySize):
    #         key[i][j] = 0
    #         for k in range(keySize):
    #             key[i][j] += matrixC[i][k] * matrixP[k][j]
    #         key[i][j] = (key[i][j] + 26) % 26

    for i in range(keySize):
        for j in range(keySize):
            key[i][j] %= 26

    return key


if __name__ == "__main__":
    main()
```

screenshot program:

Tampilan menu program

```
HILL CIPHER
1. Encrypt
2. Decrypt
3. Find Key
4. Exit
Please chose action (1/2/3): 
```

Fungsi enkripsi dengan matrix 2x2 dan 3x3

```
Please chose action (1/2/3): 1
Input plaintext: friday
input key matrix size (2/3): 2
input 0, 0 element: 7
input 0, 1 element: 8
input 1, 0 element: 19
input 1, 1 element: 3
Encrypted text: pqcfku
```

```
Please chose action (1/2/3): 1
Input plaintext: Nabil Fikri
input key matrix size (2/3): 3
input 0, 0 element: 1
input 0, 1 element: 3
input 0, 2 element: 5
input 1, 0 element: 2
input 1, 1 element: 4
input 1, 2 element: 7
input 2, 0 element: 8
input 2, 1 element: 3
input 2, 2 element: 5
Encrypted text: Shfor Sttxkjo
```

Fungsi dekripsi

```
Please chose action (1/2/3): 2
Input ciphertext: pqcfku
input key matrix size (2/3): 2
input 0, 0 element: 7
input 0, 1 element: 8
input 1, 0 element: 19
input 1, 1 element: 3
Encrypted text: friday
==============================
```

```
Please chose action (1/2/3): 2
Input ciphertext: Shfor Sttxkjo
input key matrix size (2/3): 1
matrix size can only be 2x2 or 3x3 (2/3)!
input key matrix size (2/3): 3
input 0, 0 element: 1
input 0, 1 element: 3
input 0, 2 element: 5
input 1, 0 element: 2
input 1, 1 element: 4
input 1, 2 element: 7
input 2, 0 element: 8
input 2, 1 element: 3
input 2, 2 element: 5
Encrypted text: Nabil Fikrixx
```

:. Huruf x ditambahkan pada fungsi enkripsi supaya banyak karakter dalam string berkelipatan sesuai dengan besar matrix

Fungsi mencari key

```
HILL CIPHER
1. Encrypt
2. Decrypt
3. Find Key
4. Exit
Please chose action (1/2/3): 3
input plaintext: friday
input ciphertext: pqcfku
input key matrix size (2/3): 2
[[ 7  8]
 [19  3]]
================================
```

```
HILL CIPHER
1. Encrypt
2. Decrypt
3. Find Key
4. Exit
Please chose action (1/2/3): 3
input plaintext: breathtaking
input ciphertext: rupotentoifv
input key matrix size (2/3): 3
[[ 3  4  6]
 [21 15 14]
 [20 23  5]]
================================
```

Pembuktian plaintext breathtaking

```
HILL CIPHER
1. Encrypt
2. Decrypt
3. Find Key
4. Exit
Please chose action (1/2/3): 1
Input plaintext: breathtaking
input key matrix size (2/3): 3
input 0, 0 element: 3
input 0, 1 element: 4
input 0, 2 element: 6
input 1, 0 element: 21
input 1, 1 element: 15
input 1, 2 element: 14
input 2, 0 element: 20
input 2, 1 element: 23
input 2, 2 element: 5
Encrypted text: rupotentoifv
================================
```