Nama : Ibrahim Dafi Iskandar

NPM : 140810210039

Kelas : A

Tugas 3 Praktikum Kriptografi

```
# Nama      : Ibrahim Dafi Iskandar
# NPM       : 140810210039
# Kelas     : A
# Tanggal   : 26 September 2023
# Deskripsi : Program enkripsi dan dekripsi menggunakan metode Hill Cipher

import numpy as np

def char_to_number(x):
    x = ord(x)-65
    return x

def number_to_char(x):
    x = chr(x+65)
    return x

def mod_inverse(A, M):
    for X in range(1, M):
        if (((A % M) * (X % M)) % M == 1):
            return X
    return -1

def input_key(n):
    key = list(map(int, input("Masukkan nilai key matrix (e.g 1 1 1 1):
").split()))
    key = np.array(key).reshape(n, n) % 26

    print("Key Matrix: ")
    print(key)

    return key

def input_text(string):
    text = input("Masukkan " + string + ": ")
    text = text.replace(' ', '').upper()
    return text

def hill(method, text, key, n):
    key_det = np.linalg.det(key).astype(int)

    if(len(text) % n != 0) :
        last_char = text[-1]
```

```
        text = last_char*(n - len(text) % n)

    text_in_number = list(map(char_to_number, list(text)))
    text_vector = np.array(text_in_number).reshape(int(len(text)/n), n)

    result = np.array([], dtype=int)
    if method == 'dekripsi':
        det_inverse = mod_inverse(key_det % 26, 26)
        key = (
            det_inverse * np.round(key_det * np.linalg.inv(key)).astype(int) %
26
        )

    for i in range(len(text_vector)):
            temp = np.matmul(key, text_vector[i].reshape(n, 1)) % 26
            result = np.append(result, temp)
    result = list(map(number_to_char, result))

    output = ''.join(result)

    return output

def find_key(pt, ct, m):
    pt_in_number = list(map(char_to_number, list(pt)))
    pt_vector = np.array(pt_in_number).reshape(int(len(pt)/m), m)
    p_matrix = np.array([], dtype=int)

    ct_in_number = list(map(char_to_number, list(ct)))
    ct_vector = np.array(ct_in_number).reshape(int(len(ct)/m), m)
    c_matrix = np.array([], dtype=int)

    for i in range(m):
        c_matrix = np.append(c_matrix, ct_vector[i])
        p_matrix = np.append(p_matrix, pt_vector[i])

    c_matrix = np.transpose(c_matrix.reshape(m,m))
    p_matrix = np.transpose(p_matrix.reshape(m,m))

    p_det = int(np.linalg.det(p_matrix))
    if p_det % 2 == 0 or p_det == 13 :
        print("Determinan bukan ganjil selain 13. Key tidak ada karena invers
tidak ada.")
        return

    p_det_inverse = mod_inverse(p_det % 26, 26)
    p_inverse = (
        p_det_inverse * np.round(p_det * np.linalg.inv(p_matrix)).astype(int)
% 26
```

```
    )

    key = np.matmul(c_matrix, p_inverse) % 26
    return key


while True :
    print("\n=== Program Hill Cipher ===")
    print("1. Enkripsi\n2. Dekripsi\n3. Cari Key\n4. Keluar")
    pilihan = input("Pilihan: ")

    if pilihan == '1' or pilihan == '2':
        n = int(input("\nMasukkan ukuran key matrix (n x n): "))
        key = input_key(n)
        print("\n")

        text = ''
        while(len(text) < n):
            text = input_text("text")
            if(len(text) < n):
                print("n harus bilangan prima terkecil sebagai faktor dari
jumlah karakter")

        if(pilihan == '1'):
            print("\nPlaintext: " + text)
            output = hill("enkripsi", text, key, n)
            print("Ciphertext: ", output)
        elif pilihan == '2':
            print("\nCiphertext: " + text)
            output = hill("dekripsi", text, key, n)
            print("Plaintext: ", output)

    elif pilihan == '3':
        print("\n")
        pt = input_text("plaintext")
        ct = input_text("ciphertext")
        m = int(input("\nMasukkan nilai m: "))

        print("\nPlaintext: " + pt + "\nCiphertext: " + ct)
        key = find_key(pt, ct, m)

        print("key:")
        print(key)

    elif pilihan == '4':
        exit()

    else :
```

```
        print("\nInput tidak sesuai.\n")
```

Penjelasan kode:

def char_to_number(*x*) => fungsi untuk mengubah char ke angka yang bersesuaian

def number_to_char(x) => fungsi untuk mengubah angka ke char yang bersesuaian

def mod_inverse(*A*, *M*) => Fungsi untuk mencari modular multiplicative inverse dari suatu angka

def input_key(*n*) => Fungsi untuk menginput key matrix

def hill(*method*, *text*, *key*, *n*) => Fungsi untuk melakukan operasi hill cipher

def find_key(*pt*, *ct*, *m*) => Fungsi untuk mencari key dari cipher text hill cipher

Main menu => menu untuk memilih fitur program