Nama : Vernandika Stanley Hansen
NPM : 140810220031
Praktikum Kriptografi
Hill Cipher

Tugas 2

Program Hill Cipher

```
import math
import string
import sys
import numpy as np
from sympy import Matrix

#menu
def menu():
    while True:
        print("----Program Hill Cipher----")
        print("1) Enkripsi")
        print("2) Dekripsi")
        print("3) Mencari Kunci")
        print("4) Keluar\n")
        try:
            choice = int(input("Pilih: "))
            if 1 <= choice <= 4:
                return choice
            else:
                print("\nMasukkan angka 1-4\n")
        except ValueError:
            print("\nMasukkan angka 1-4\n")

# alfabet ke angka, angka ke alfabet
def get_alphabet():
    alphabet = {}
    for character in string.ascii_uppercase:
        alphabet[character] = string.ascii_uppercase.index(character)

    reverse_alphabet = {}
    for key, value in alphabet.items():
        reverse_alphabet[value] = key
```

```
    return alphabet, reverse_alphabet

# input harus huruf alfabet
def get_text_input(message, alphabet):
    while True:
        text = input(message)
        text = text.upper()
        if all(keys in alphabet for keys in text):
            return text
        else:
            print("\nHanya masukkan huruf [a-z,A-Z]")

# cek apakah matriks berbentuk persegi untuk kunci
def is_square(key):
    key_length = len(key)
    if 2 <= key_length == int(math.sqrt(key_length)) ** 2:
        return True
    else:
        return False

# matriks k untuk kunci
def get_key_matrix(key):
    k = list(map(int, key.split()))

    m = int(math.sqrt(len(k)))
    if m ** 2 != len(k):
        raise ValueError("Panjang kunci harus menghasilkan matriks
persegi")

    return np.reshape(k, (m, m))

# mengambil bentuk matriks menjadi huruf alfabet
def get_text_matrix(text, m, alphabet):
    matrix = list(text)
    remainder = len(text) % m
    for (i, character) in enumerate(matrix):
        matrix[i] = alphabet[character]
    if remainder != 0:
        for i in range(m - remainder):
```

```
            matrix.append(25)

    return np.reshape(matrix, (int(len(matrix) / m), m)).transpose()

# fungsi enkripsi hill cipher
def encrypt(key, plaintext, alphabet):
    m = key.shape[0]
    m_grams = plaintext.shape[1]

    ciphertext = np.zeros((m, m_grams)).astype(int)
    for i in range(m_grams):
        ciphertext[:, i] = np.reshape(np.dot(key, plaintext[:, i]) %
len(alphabet), m)
    return ciphertext

# mengubah bentuk matriks menjadi teks
def matrix_to_text(matrix, order, alphabet):
    if order == 't':
        text_array = np.ravel(matrix, order='F')
    else:
        text_array = np.ravel(matrix)
    text = ""
    for i in range(len(text_array)):
        text = text + alphabet[text_array[i]]
    return text

# fungsi invers matriks
def get_inverse(matrix, alphabet):
    alphabet_len = len(alphabet)
    if math.gcd(int(round(np.linalg.det(matrix))), alphabet_len) == 1:
        matrix = Matrix(matrix)
        return np.matrix(matrix.inv_mod(alphabet_len))
    else:
        return None

# fungsi dekripsi
def decrypt(k_inverse, c, alphabet):
    return encrypt(k_inverse, c, alphabet)

# mengambil nilai m (ukuran matriks persegi)
```

```
def get_m():
    while True:
        try:
            m = int(input("masukkan m (ukuran matriks persegi): "))
            if m >= 2:
                return m
            else:
                print("\nnilai harus lebih dari sama dengan 2\n")
        except ValueError:
            print("\nnilai harus lebih dari sama dengan 2\n")

# fungsi mencari kunci
def find_key(c, p_inverse, alphabet):
    return encrypt(c, p_inverse, alphabet)

#fungsi utama
def main():
    while True:
        choice = menu()

        alphabet, reverse_alphabet = get_alphabet()

        if choice == 1:
            plaintext = get_text_input("\nMasukkan Plain Text: ",
alphabet)
            key = input("Masukkan kunci (tiap angka pisahkan dengan
spasi): ")

            try:
                k = get_key_matrix(key)
                print("\nMatriks Kunci:\n", k)

                p = get_text_matrix(plaintext, k.shape[0], alphabet)
                print("Matriks Plain Text :\n", p)

                c = encrypt(k, p, alphabet)

                ciphertext = matrix_to_text(c, "t", reverse_alphabet)

                print("\nHasil Enkripsi\n")
```

```
                print("Matriks Cipher Text:\n", c, "\n")
                print("Cipher Text: ", ciphertext)
            except ValueError as e:
                print("\nError:", e)

        elif choice == 2:
            ciphertext = get_text_input("\nMasukkan Cipher Text: ",
alphabet)
            key = input("Masukkan kunci (tiap angka pisahkan dengan spasi)
")

            try:
                k = get_key_matrix(key)

                k_inverse = get_inverse(k, alphabet)

                if k_inverse is not None:
                    c = get_text_matrix(ciphertext, k_inverse.shape[0],
alphabet)


                    print("\nMatriks Kunci:\n", k)
                    print("Matriks Cipher Text:\n", c)

                    p = decrypt(k_inverse, c, alphabet)

                    plaintext = matrix_to_text(p, "t", reverse_alphabet)

                    print("\nHasil Dekripsi\n")
                    print("Matriks Plaintext:\n", p, "\n")
                    print("Plaintext: ", plaintext)
                else:
                    print("\nMatriks tidak dapat didekripsi\n")
            except ValueError as e:
                print("\nError:", e)

        elif choice == 3:
            plaintext = get_text_input("\nMasukkan Plain Text: ",
alphabet)
            ciphertext = get_text_input("Masukkan Cipher Text: ",
alphabet)
```

```
            m = get_m()

            if len(plaintext) / m >= m:
                p = get_text_matrix(plaintext, m, alphabet)
                p = p[:, 0:m]

                p_inverse = get_inverse(p, alphabet)

                if p_inverse is not None:
                    c = get_text_matrix(ciphertext, m, alphabet)
                    c = c[:, 0:m]

                    if c.shape[1] == p.shape[0]:
                        print("\nMatriks Cipher Text:\n", c)
                        print("Matriks Plain Text:\n", p)

                        k = find_key(c, p_inverse, alphabet)

                        key = matrix_to_text(k, "k", reverse_alphabet)

                        print("\nHasil Kunci\n")
                        print("Matriks Kunci:\n", k, "\n")
                        # print("Kunci: ", key)
                    else:
                        print("\nTUkuran Plain Text dan Cipher Text
berbeda\n")
                else:
                    print("\nMatriks tidak dapat diubah\n")
            else:
                print("\nUkuran Plain Text harus kompatibel dengan ukuran
matriks kunci\n")
        elif choice == 4:
            sys.exit(0)


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

Penjelasan

1. Import library library yang dibutuhkan untuk program

```
import math
import string
import sys
import numpy as np
from sympy import Matrix
```

2. Fungsi untuk menu

```
def menu():
    while True:
        print("----Program Hill Cipher----")
        print("1) Enkripsi")
        print("2) Dekripsi")
        print("3) Mencari Kunci")
        print("4) Keluar\n")
        try:
            choice = int(input("Pilih: "))
            if 1 <= choice <= 4:
                return choice
            else:
                print("\nMasukkan angka 1-4\n")
        except ValueError:
            print("\nMasukkan angka 1-4\n")
```

3. Fungsi untuk mengubah huruf alfabet ke angka dan angka ke alfabet misalnya 0 => A dan Z = 25

```
# alfabet ke angka, angka ke alfabet
def get_alphabet():
    alphabet = {}
    for character in string.ascii_uppercase:
        alphabet[character] = string.ascii_uppercase.index(character)

    reverse_alphabet = {}
    for key, value in alphabet.items():
        reverse_alphabet[value] = key

    return alphabet, reverse_alphabet
```

4. Fungsi ini memastikan bahwa nilai yang diinput berupa huruf alfabet

```
def get_text_input(message, alphabet):
    while True:
        text = input(message)
        text = text.upper()
        if all(keys in alphabet for keys in text):
            return text
        else:
            print("\nHanya masukkan huruf [a-z,A-Z]")
```

5. Fungsi ini memastikan bahwa matriks yang diinput berukuran persegi (nxn)

```
def is_square(key):
    key_length = len(key)
    if 2 <= key_length == int(math.sqrt(key_length)) ** 2:
        return True
    else:
        return False
```

6. Fungsi untuk mengambil nilai matriks kunci dan fungsi untuk mengambil bentuk matriks menjadi alfabet

```
def get_key_matrix(key):
    k = list(map(int, key.split()))

    m = int(math.sqrt(len(k)))
    if m ** 2 != len(k):
        raise ValueError("Panjang kunci harus menghasilkan matriks
persegi")

    return np.reshape(k, (m, m))

def get_text_matrix(text, m, alphabet):
    matrix = list(text)
    remainder = len(text) % m
    for (i, character) in enumerate(matrix):
        matrix[i] = alphabet[character]
    if remainder != 0:
        for i in range(m - remainder):
            matrix.append(25)

    return np.reshape(matrix, (int(len(matrix) / m), m)).transpose()
```

7. Fungsi rumus enkripsi

```
def encrypt(key, plaintext, alphabet):
    m = key.shape[0]
    m_grams = plaintext.shape[1]

    ciphertext = np.zeros((m, m_grams)).astype(int)
    for i in range(m_grams):
        ciphertext[:, i] = np.reshape(np.dot(key, plaintext[:, i]) %
len(alphabet), m)
    return ciphertext
```

8. Fungsi mengubah bentuk matriks menjadi huruf alfabet

```
def matrix_to_text(matrix, order, alphabet):
    if order == 't':
        text_array = np.ravel(matrix, order='F')
    else:
        text_array = np.ravel(matrix)
    text = ""
    for i in range(len(text_array)):
        text = text + alphabet[text_array[i]]
    return text
```

9. Fungsi untuk invers matriks

```
def get_inverse(matrix, alphabet):
    alphabet_len = len(alphabet)
    if math.gcd(int(round(np.linalg.det(matrix))), alphabet_len) == 1:
        matrix = Matrix(matrix)
        return np.matrix(matrix.inv_mod(alphabet_len))
    else:
        return None
```

10. Fungsi dekripsi dan mencari kunci menggunakan fungsi enkripsi hanya mengubah parameter saja

```
def decrypt(k_inverse, c, alphabet):
    return encrypt(k_inverse, c, alphabet)
def find_key(c, p_inverse, alphabet):
    return encrypt(c, p_inverse, alphabet)
```

11. Fungsi untuk mengambil nilai m dan memastikan bahwa nilai nya lebih dari sama dengan 2

```
def get_m():
    while True:
        try:
            m = int(input("masukkan m (ukuran matriks persegi): "))
            if m >= 2:
                return m
            else:
                print("\nnilai harus lebih dari sama dengan 2\n")
        except ValueError:
            print("\nnilai harus lebih dari sama dengan 2\n")
```

12. Fungsi utama

    Jika memilih pilihan 1 maka akan diarahkan pada input input yang dibutuhkan untuk proses enkripsi yaitu plain text dan kunci nya baru akan menghasilkan cipher text menggunakan hill cipher. Jika memilih pilihan 2 akan diarahkan ke proses dekripsi dan meminta cipher text dan kunci. Sedangkan jika pilihan 3 maka akan diarahkan ke pilihan mencari kunci dan diminta plain text, cipher text, dan ukuran matriks kunci. Terakhir jika 4 maka akan keluar program. Pilihan selain itu akan me looping menu terus terusan.

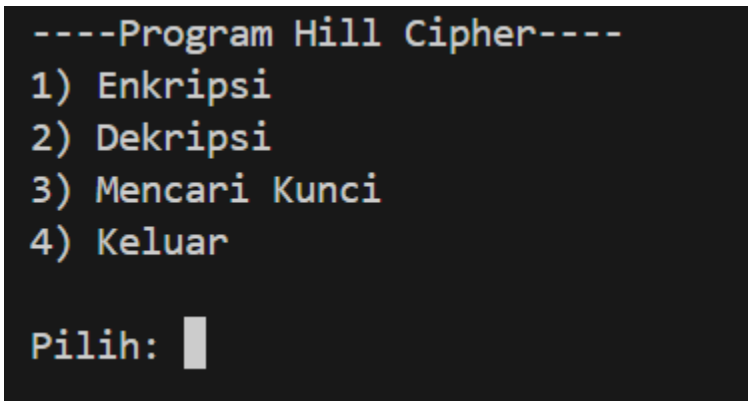
```
def main():
    while True:
        choice = menu()

        alphabet, reverse_alphabet = get_alphabet()

        if choice == 1:
            plaintext = get_text_input("\nMasukkan Plain Text: ",
alphabet)
            key = input("Masukkan kunci (tiap angka pisahkan dengan
spasi): ")

            try:
                k = get_key_matrix(key)
                print("\nMatriks Kunci:\n", k)

                p = get_text_matrix(plaintext, k.shape[0], alphabet)
                print("Matriks Plain Text :\n", p)

                c = encrypt(k, p, alphabet)

                ciphertext = matrix_to_text(c, "t", reverse_alphabet)
```

```
                print("\nHasil Enkripsi\n")
                print("Matriks Cipher Text:\n", c, "\n")
                print("Cipher Text: ", ciphertext)
            except ValueError as e:
                print("\nError:", e)

        elif choice == 2:
            ciphertext = get_text_input("\nMasukkan Cipher Text: ",
alphabet)
            key = input("Masukkan kunci (tiap angka pisahkan dengan spasi)
")

            try:
                k = get_key_matrix(key)

                k_inverse = get_inverse(k, alphabet)

                if k_inverse is not None:
                    c = get_text_matrix(ciphertext, k_inverse.shape[0],
alphabet)

                    print("\nMatriks Kunci:\n", k)
                    print("Matriks Cipher Text:\n", c)

                    p = decrypt(k_inverse, c, alphabet)

                    plaintext = matrix_to_text(p, "t", reverse_alphabet)

                    print("\nHasil Dekripsi\n")
                    print("Matriks Plaintext:\n", p, "\n")
                    print("Plaintext: ", plaintext)
                else:
                    print("\nMatriks tidak dapat didekripsi\n")
            except ValueError as e:
                print("\nError:", e)

        elif choice == 3:
            plaintext = get_text_input("\nMasukkan Plain Text: ",
alphabet)
```

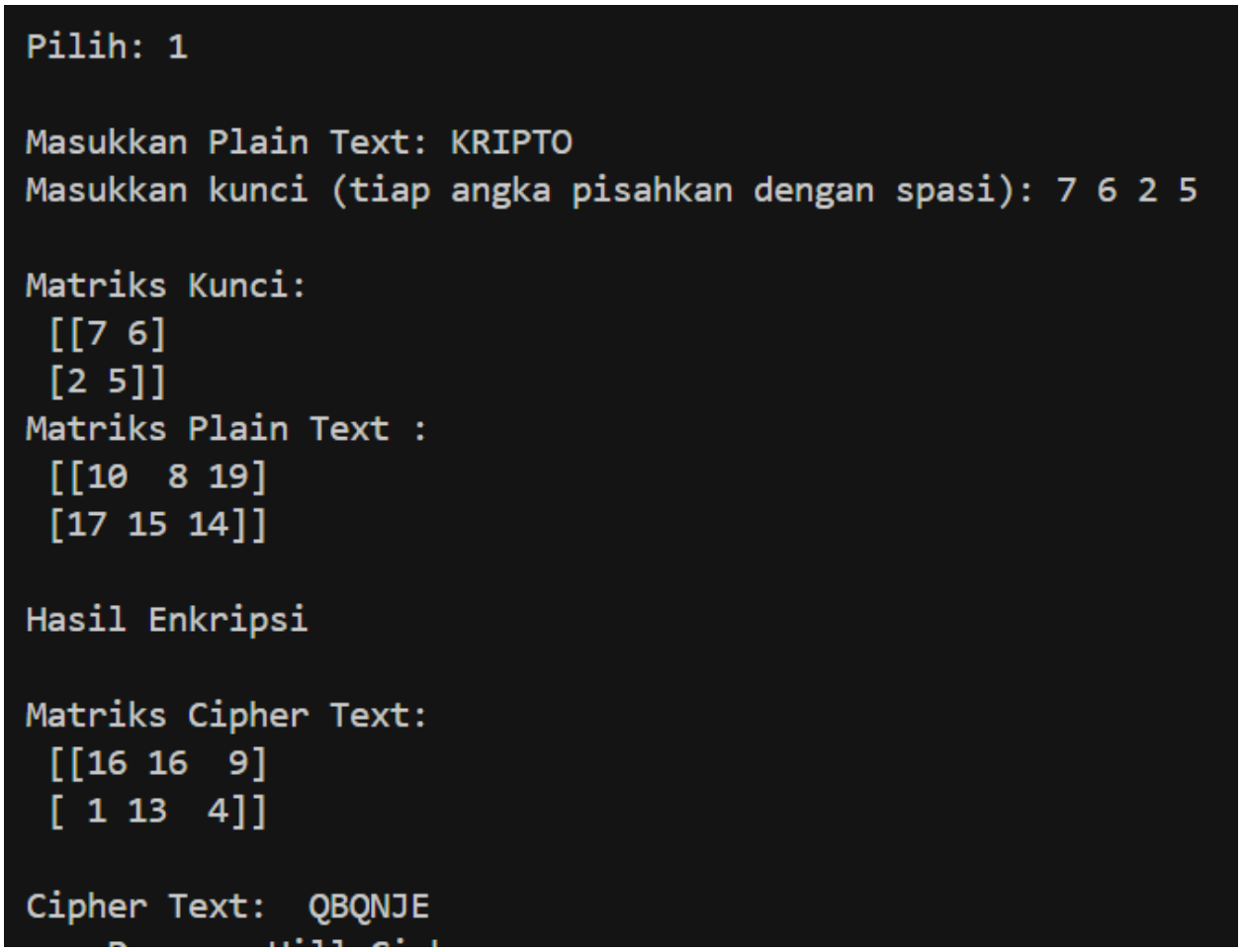
```
            ciphertext = get_text_input("Masukkan Cipher Text: ",
alphabet)

            m = get_m()

            if len(plaintext) / m >= m:
                p = get_text_matrix(plaintext, m, alphabet)
                p = p[:, 0:m]

                p_inverse = get_inverse(p, alphabet)

                if p_inverse is not None:
                    c = get_text_matrix(ciphertext, m, alphabet)
                    c = c[:, 0:m]

                    if c.shape[1] == p.shape[0]:
                        print("\nMatriks Cipher Text:\n", c)
                        print("Matriks Plain Text:\n", p)

                        k = find_key(c, p_inverse, alphabet)

                        key = matrix_to_text(k, "k", reverse_alphabet)

                        print("\nHasil Kunci\n")
                        print("Matriks Kunci:\n", k, "\n")
                        # print("Kunci: ", key)
                    else:
                        print("\nTUkuran Plain Text dan Cipher Text
berbeda\n")
                else:
                    print("\nMatriks tidak dapat diubah\n")
            else:
                print("\nUkuran Plain Text harus kompatibel dengan ukuran
matriks kunci\n")
        elif choice == 4:
            sys.exit(0)
```

Hasil SS Program

Menu



Enkripsi



Dekripsi

```
Pilih: 2

Masukkan Cipher Text: QBQNJE
Masukkan kunci (tiap angka pisahkan dengan spasi) 7 6 2 5

Matriks Kunci:
 [[7 6]
 [2 5]]
Matriks Cipher Text:
 [[16 16  9]
 [ 1 13  4]]

Hasil Dekripsi

Matriks Plaintext:
 [[10  8 19]
 [17 15 14]]

Plaintext:  KRIPTO
```

Mencari Kunci

```
Pilih: 3

Masukkan Plain Text: FRIDAY
Masukkan Cipher Text: PQCFKU
masukkan m (ukuran matriks persegi): 2

Matriks Cipher Text:
 [[15  2]
 [16  5]]
Matriks Plain Text:
 [[ 5  8]
 [17  3]]

Hasil Kunci

Matriks Kunci:
 [[ 7  8]
 [19  3]]
```

Keluar Program

```
 ----Program Hill Cipher----
 1) Enkripsi
 2) Dekripsi
 3) Mencari Kunci
 4) Keluar

 Pilih: 4
 PS D:\sem_5\prak_kripto\31-Kripto24\Hill-Cipher>
```