Nama : Muhammad Adzikra Dhiya Alfauzan
NPM : 140810220046

```
#include <iostream>

using namespace std;

const int MOD = 26;

void multiplyMatrix(int key[2][2], int textVec[2], int result[2]) {
    result[0] = (key[0][0] * textVec[0] + key[0][1] * textVec[1]) %
MOD;
    result[1] = (key[1][0] * textVec[0] + key[1][1] * textVec[1]) %
MOD;
}

int determinant(int matrix[2][2]) {
    return (matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0])
% MOD;
}

int modInverse(int a, int mod) {
    a = a % mod;
    for (int x = 1; x < mod; x++) {
        if ((a * x) % mod == 1)
            return x;
    }
    return -1;
}

void inverseMatrix(int matrix[2][2], int invMatrix[2][2]) {
    int det = determinant(matrix);
    int invDet = modInverse(det, MOD);

    if (invDet == -1) {
        cout << "Invers tidak ditemukan (matriks singular)" << endl;
        exit(1);
    }

    invMatrix[0][0] = (matrix[1][1] * invDet) % MOD;
    invMatrix[0][1] = (-matrix[0][1] * invDet) % MOD;
    invMatrix[1][0] = (-matrix[1][0] * invDet) % MOD;
    invMatrix[1][1] = (matrix[0][0] * invDet) % MOD;
```

```
    for (int i = 0; i < 2; i++)
        for (int j = 0; j < 2; j++)
            invMatrix[i][j] = (invMatrix[i][j] + MOD) % MOD;
}

string encrypt(string plaintext, int key[2][2]) {
    string ciphertext = "";
    for (int i = 0; i < plaintext.length(); i += 2) {
        int textVec[2] = { plaintext[i] - 'A', plaintext[i + 1] - 'A'
};
        int result[2];
        multiplyMatrix(key, textVec, result);
        ciphertext += (result[0] + 'A');
        ciphertext += (result[1] + 'A');
    }
    return ciphertext;
}

string decrypt(string ciphertext, int key[2][2]) {
    int invKey[2][2];
    inverseMatrix(key, invKey);
    string plaintext = "";
    for (int i = 0; i < ciphertext.length(); i += 2) {
        int textVec[2] = { ciphertext[i] - 'A', ciphertext[i + 1] - 'A'
};
        int result[2];
        multiplyMatrix(invKey, textVec, result);
        plaintext += (result[0] + 'A');
        plaintext += (result[1] + 'A');
    }
    return plaintext;
}

void findKey(int plaintext[2][2], int ciphertext[2][2], int key[2][2])
{
    int invPlaintext[2][2];
    inverseMatrix(plaintext, invPlaintext);

    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            key[i][j] = (ciphertext[i][0] * invPlaintext[0][j] +
ciphertext[i][1] * invPlaintext[1][j]) % MOD;
```

```
            if (key[i][j] < 0) key[i][j] += MOD;
        }
    }
}

void inputMatrix(int matrix[2][2], string label) {
    cout << "Masukkan matriks " << label << " (2x2, nilai harus antara
0-25):" << endl;
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            cout << "Elemen [" << i << "][" << j << "]: ";
            cin >> matrix[i][j];
        }
    }
}

void menu() {
    int key[2][2], plaintextMatrix[2][2], ciphertextMatrix[2][2];
    string plaintext, ciphertext;
    int choice;

    do {
        cout << "\n=== Menu Hill Cipher ===" << endl;
        cout << "1. Enkripsi (input kunci dan plaintext)" << endl;
        cout << "2. Dekripsi (input kunci dan ciphertext)" << endl;
        cout << "3. Mencari kunci (input plaintext dan ciphertext)" <<
endl;
        cout << "4. Keluar" << endl;
        cout << "Pilih opsi: ";
        cin >> choice;

        switch (choice) {
            case 1:
                inputMatrix(key, "kunci");
                cout << "Masukkan teks untuk dienkripsi (jumlah
karakter genap, huruf A-Z): ";
                cin >> plaintext;
                ciphertext = encrypt(plaintext, key);
                cout << "Hasil enkripsi (ciphertext): " << ciphertext
<< endl;
                break;

            case 2:
```

```
                inputMatrix(key, "kunci");
                cout << "Masukkan teks untuk didekripsi (ciphertext):
";
                cin >> ciphertext;
                plaintext = decrypt(ciphertext, key);
                cout << "Hasil dekripsi (plaintext): " << plaintext <<
endl;
                break;

            case 3:
                cout << "Masukkan plaintext (4 huruf pertama): ";
                cin >> plaintext;
                cout << "Masukkan ciphertext (4 huruf pertama): ";
                cin >> ciphertext;

                plaintextMatrix[0][0] = plaintext[0] - 'A';
                plaintextMatrix[0][1] = plaintext[1] - 'A';
                plaintextMatrix[1][0] = plaintext[2] - 'A';
                plaintextMatrix[1][1] = plaintext[3] - 'A';

                ciphertextMatrix[0][0] = ciphertext[0] - 'A';
                ciphertextMatrix[0][1] = ciphertext[1] - 'A';
                ciphertextMatrix[1][0] = ciphertext[2] - 'A';
                ciphertextMatrix[1][1] = ciphertext[3] - 'A';

                findKey(plaintextMatrix, ciphertextMatrix, key);

                cout << "Kunci yang ditemukan adalah:" << endl;
                for (int i = 0; i < 2; i++) {
                    for (int j = 0; j < 2; j++) {
                        cout << key[i][j] << " ";
                    }
                    cout << endl;
                }
                break;

            case 4:
                cout << "Keluar dari program..." << endl;
                break;

            default:
                cout << "Opsi tidak valid, silakan coba lagi." << endl;
        }
```

```
    } while (choice != 4);
}

int main() {
    menu();
    return 0;
}
```

Hasil

```
PS D:\File\SEMESTER 5\TUGAS\Kriptografi\Praktikum\46-Kripto24\Hill-Cipher> ./hillcipher

=== Menu Hill Cipher ===
1. Enkripsi (input kunci dan plaintext)
2. Dekripsi (input kunci dan ciphertext)
3. Mencari kunci (input plaintext dan ciphertext)
4. Keluar
Pilih opsi: 1
Masukkan matriks kunci (2x2, nilai harus antara 0-25):
Elemen [0][0]: 7
Elemen [0][1]: 6
Elemen [1][0]: 2
Elemen [1][1]: 5
Masukkan teks untuk dienkripsi (jumlah karakter genap, huruf A-Z): PYTHON
Hasil enkripsi (ciphertext): PUTVUP

=== Menu Hill Cipher ===
1. Enkripsi (input kunci dan plaintext)
2. Dekripsi (input kunci dan ciphertext)
3. Mencari kunci (input plaintext dan ciphertext)
4. Keluar
Pilih opsi: 2
Masukkan matriks kunci (2x2, nilai harus antara 0-25):
Elemen [0][0]: 7
Elemen [0][1]: 6
Elemen [1][0]: 2
Elemen [1][1]: 5
Masukkan teks untuk didekripsi (ciphertext): PUTVUP
Hasil dekripsi (plaintext): PYTHON
```

```
=== Menu Hill Cipher ===
1. Enkripsi (input kunci dan plaintext)
2. Dekripsi (input kunci dan ciphertext)
3. Mencari kunci (input plaintext dan ciphertext)
4. Keluar
Pilih opsi: 3
Masukkan plaintext (4 huruf pertama): TEST
Masukkan ciphertext (4 huruf pertama): ZSFL
Kunci yang ditemukan adalah:
7 20
9 11
```

Penjelasan :

1. Function
    - **multiplyMatrix**: Mengalikan matriks kunci dengan vektor teks (plaintext atau ciphertext), lalu hasilnya diambil mod 26 untuk mendapatkan huruf yang valid dalam alfabet (A-Z).
    - **determinant**: Menghitung determinan dari matriks 2x2, yang penting untuk menghitung invers matriks.
    - **modInverse**: Mencari invers modulo dari determinan matriks dengan mod 26, yang digunakan dalam pembentukan invers matriks.
    - **inverseMatrix**: Menghitung invers dari matriks kunci yang digunakan saat melakukan dekripsi.
    - **encrypt**: Mengenkripsi plaintext dengan menggunakan kunci matriks. Teks diproses dalam pasangan dua karakter, lalu dienkripsi dengan matriks kunci menggunakan fungsi multiplyMatrix.
    - **decrypt**: Mendekripsi ciphertext dengan cara mengalikan ciphertext dengan invers dari matriks kunci, menggunakan proses yang mirip dengan enkripsi.
    - **findKey**: Fungsi ini mencari kunci Hill Cipher dengan menggunakan plaintext dan ciphertext. Program menghitung invers dari matriks plaintext, kemudian mengalikannya dengan matriks ciphertext untuk menemukan kunci.
    - **inputMatrix**: Fungsi untuk meminta input matriks 2x2.
    - **menu**: Fungsi yang menyediakan menu untuk memilih antara enkripsi, dekripsi, dan pencarian kunci.
        - *Enkripsi*: Input kunci dan plaintext, lalu plaintext dienkripsi menjadi ciphertext.
        - *Dekripsi*: Input kunci dan ciphertext, lalu ciphertext didekripsi menjadi plaintext.
        - *Mencari Kunci*: Input 4 huruf plaintext dan ciphertext, lalu program menghitung kunci Hill Cipher yang digunakan untuk mengubah plaintext menjadi ciphertext.
        - *Keluar*: Keluar dari program.