Nama : Muhamad Davio Athallah
NPM : 140810200063

Penjelasan Program

```
string encrypt(string plainTeks, int n)
{
    int plaintext[1000][3] = {0};
    int ciphertext[1000][3] = {0};
    int ptloop = 0;

    while (plainTeks.length() % n != 0)
    {
        plainTeks += "x";
    }
    int row = (plainTeks.length()) / n;

    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < n; j++)
        {
            plaintext[i][j] = plainTeks[ptloop++] - 'a';
        }
    }

    multiplyMatrix(plaintext, row, n, key, n, n, ciphertext);

    string cipherTeks = "";
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < n; j++)
        {
            cipherTeks += (ciphertext[i][j] + 'a');
        }
    }
    return cipherTeks;
}
```

Fungsi encrypt untuk mengenkripsi plainteks dengan input plaintext dan ordo matriks, menggunakan fungsi multiplyMatrix

```
void multiplyMatrix(int m1[1000][3], int m1_rows, int m1_cols, int m2[1000][3], int m2_rows, int m2_cols, int hasil[1000][3])
{
    for (int i = 0; i < m1_rows; i++)
    {
        for (int j = 0; j < m2_cols; j++)
        {
            for (int k = 0; k < m2_rows; k++)
            {
                hasil[i][j] += a[i][k] * b[k][j];
            }
            hasil[i][j] = mod26(hasil[i][j]);
        }
    }
}
```

Fungsi multiplyMatrix untuk mengkalikan blok plaintext dengan matriks kunci

```
string decrypt(string cipherTeks, int n)
{
    int plaintext[1000][3] = {0};
    int ciphertext[1000][3] = {0};
    int ctloop = 0;

    int row = cipherTeks.length() / n;

    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < n; j++)
        {
            ciphertext[i][j] = cipherTeks[ctloop++] - 'a';
        }
    }

    int k_inverse[3][3] = {0};
    findInverse(key, n, k_inverse);

    multiplyMatrix(ciphertext, row, n, k_inverse, n, n, plaintext);

    string plainTeks = "";
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < n; j++)
        {
            plainTeks += (plaintext[i][j] + 'a');
        }
    }
    return plainTeks;
}
```

Fungsi decrypt untuk mendekripsi ciphertext dengan input ciphertext dan ordo matriks, menggunakan fungsi findInverse dan multiplyMatrix

```
void findInverse(int m[3][3], int n, int m_inverse[3][3])
{
    int adj[3][3] = {0};

    int det = findDet(m, n);
    int detInverse = findDetInverse(det);

    if (n == 2)
    {
        adj[0][0] = m[1][1];
        adj[1][1] = m[0][0];
        adj[0][1] = -m[0][1];
        adj[1][0] = -m[1][0];
    }

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            m_inverse[i][j] = mod26(adj[i][j] * detInverse);
        }
    }
}

int key[3][3];
```

Fungsi findInverse untuk mencari invers matriks K

```
void multiplyMatrix(int m1[1000][3], int m1_rows, int m1_cols, int m2[1000][3], int m2_rows, int m2_cols, int hasil[1000][3])
{
    for (int i = 0; i < m1_rows; i++)
    {
        for (int j = 0; j < m2_cols; j++)
        {
            for (int k = 0; k < m2_rows; k++)
            {
                hasil[i][j] += a[i][k] * b[k][j];
            }
            hasil[i][j] = mod26(hasil[i][j]);
        }
    }
}
```

Fungsi multiplyMatrix untuk mengkalikan blok ciphertext dengan matriks kunci

```
void findKey()
{
    string plainteks, cipherteks;
    int key[2][2], det, detInv, adj[2][2], plainteksInv[2][2], plainMatrix[2][2], cipMatrix[2][2], counter;
    int p, c;
    int transpose[2][2];
    cout << "Masukkan Plainteks : ";
    cin.ignore();
    getline(cin, plainteks);

    counter = 0;
    for (int i = 0; i < 2; i++)
    {
        for (int j = 0; j < 2; j++)
        {
            p = toupper(plainteks[counter]) - 65;
            plainMatrix[i][j] = p;
            counter++;
        }
    }

    cout << "Masukkan Cipherteks : ";
    getline(cin, cipherteks);
```

Fungsi findKey untuk mencari K dengan input ciphertext dan plaintext