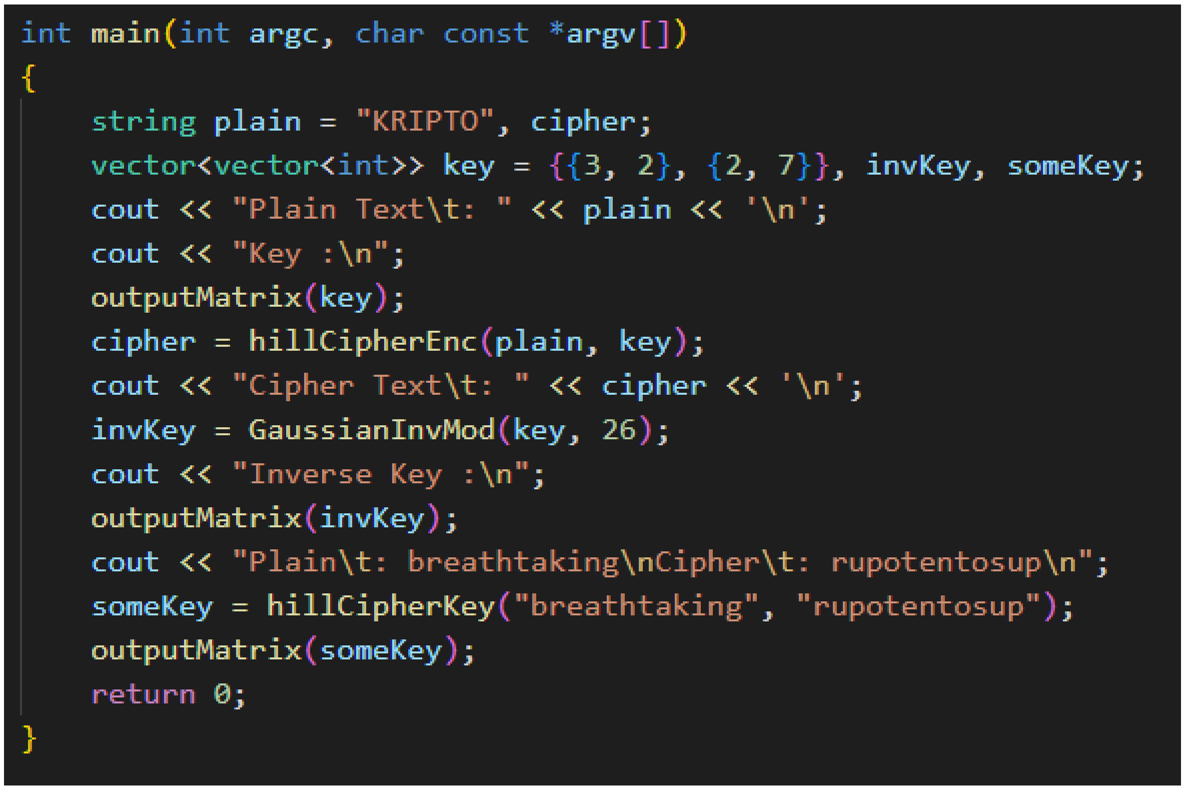
Mu’az Abdul Rohim

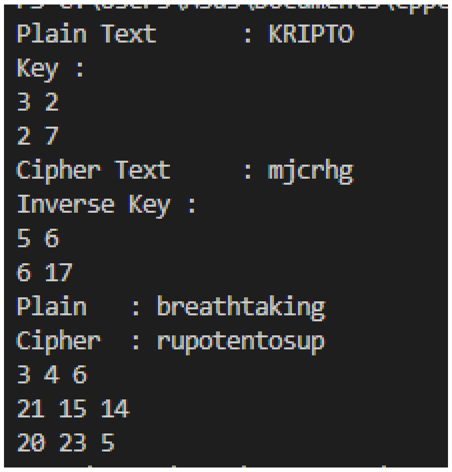
140810200026/B

Main Program



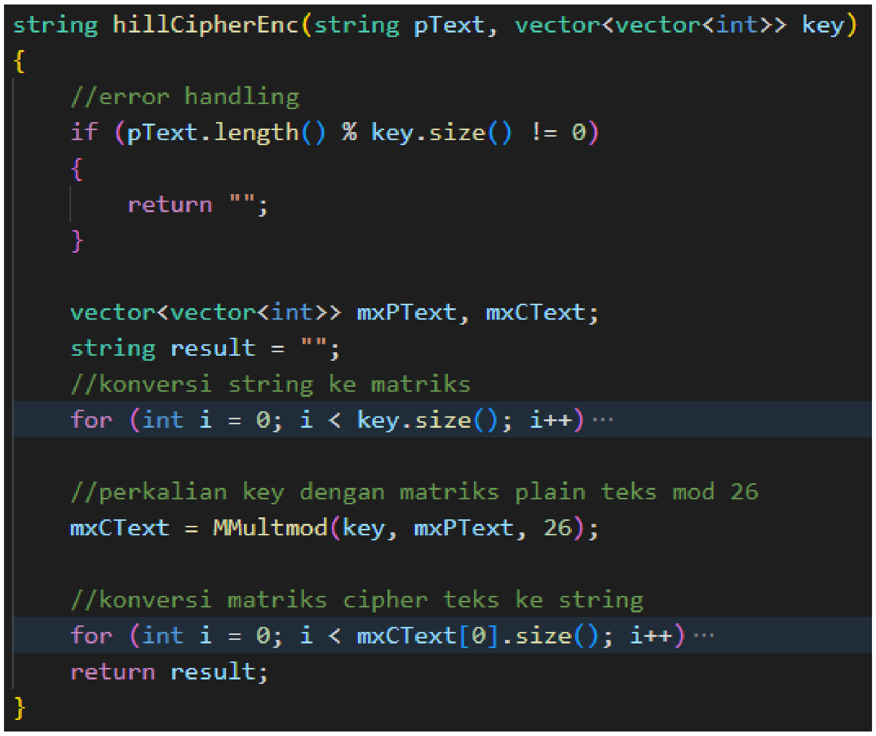
```cpp
int main(int argc, char const *argv[])
{
    string plain = "KRIPTO", cipher;
    vector<vector<int>> key = {{3, 2}, {2, 7}}, invKey, someKey;
    cout << "Plain Text\t: " << plain << '\n';
    cout << "Key :\n";
    outputMatrix(key);
    cipher = hillCipherEnc(plain, key);
    cout << "Cipher Text\t: " << cipher << '\n';
    invKey = GaussianInvMod(key, 26);
    cout << "Inverse Key :\n";
    outputMatrix(invKey);
    cout << "Plain\t: breathtaking\nCipher\t: rupotentosup\n";
    someKey = hillCipherKey("breathtaking", "rupotentosup");
    outputMatrix(someKey);
    return 0;
}
```

Hasil Program



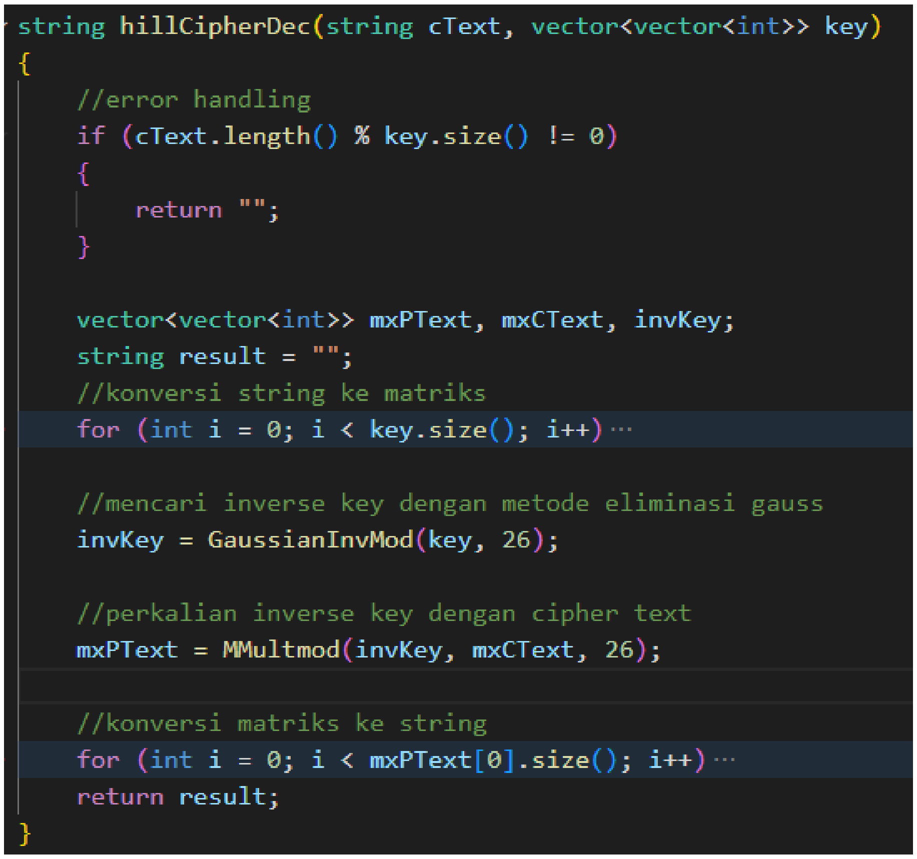
```
Plain Text      : KRIPTO
Key :
3 2
2 7
Cipher Text     : mjcrhg
Inverse Key :
5 6
6 17
Plain   : breathtaking
Cipher  : rupotentosup
3 4 6
21 15 14
20 23 5
```

Fungsi Enkripsi



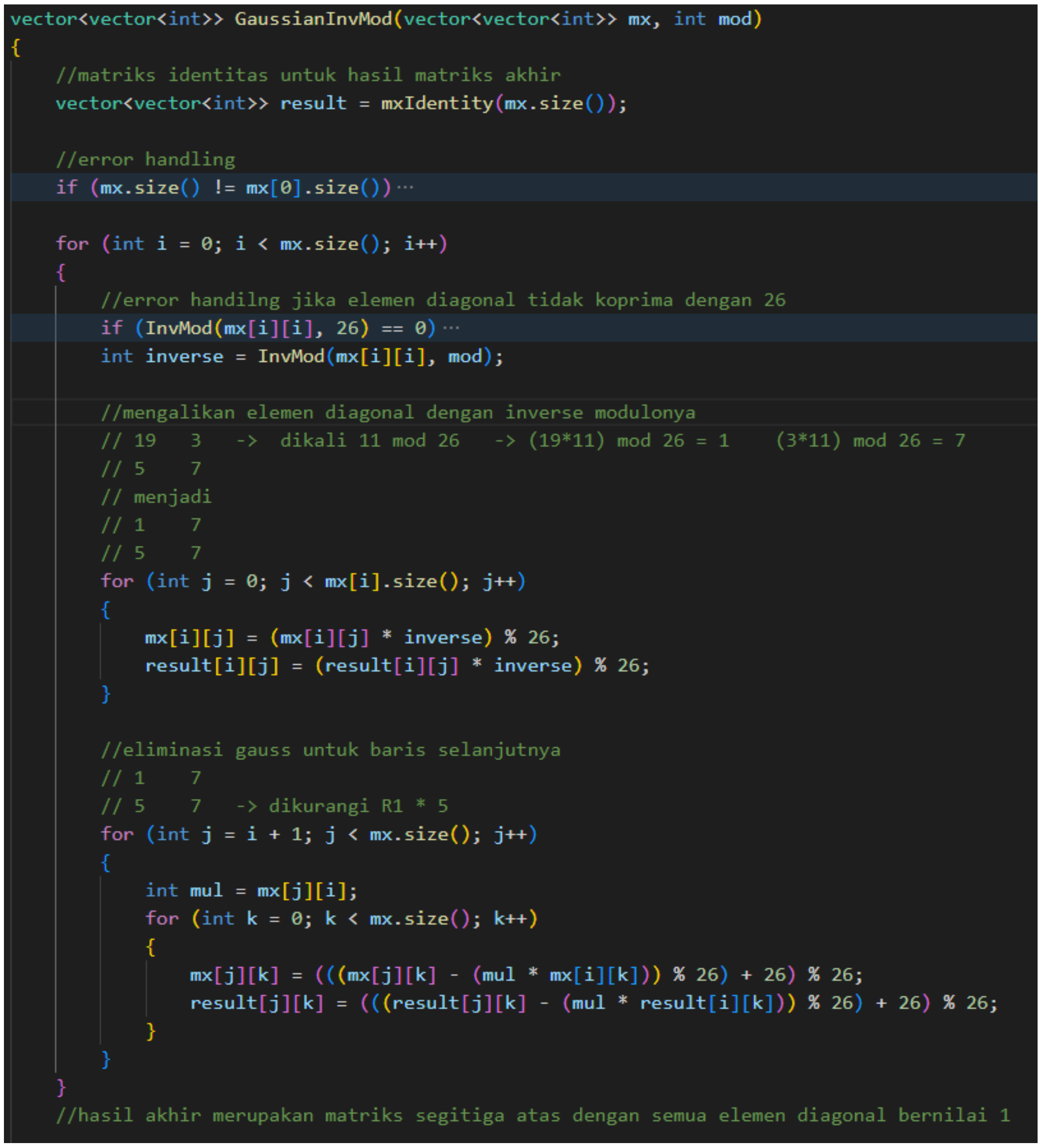
```cpp
string hillCipherEnc(string pText, vector<vector<int>> key)
{
    //error handling
    if (pText.length() % key.size() != 0)
    {
        return "";
    }

    vector<vector<int>> mxPText, mxCText;
    string result = "";
    //konversi string ke matriks
    for (int i = 0; i < key.size(); i++)···

    //perkalian key dengan matriks plain teks mod 26
    mxCText = MMultmod(key, mxPText, 26);

    //konversi matriks cipher teks ke string
    for (int i = 0; i < mxCText[0].size(); i++)···
    return result;
}
```

Fungsi Dekripsi

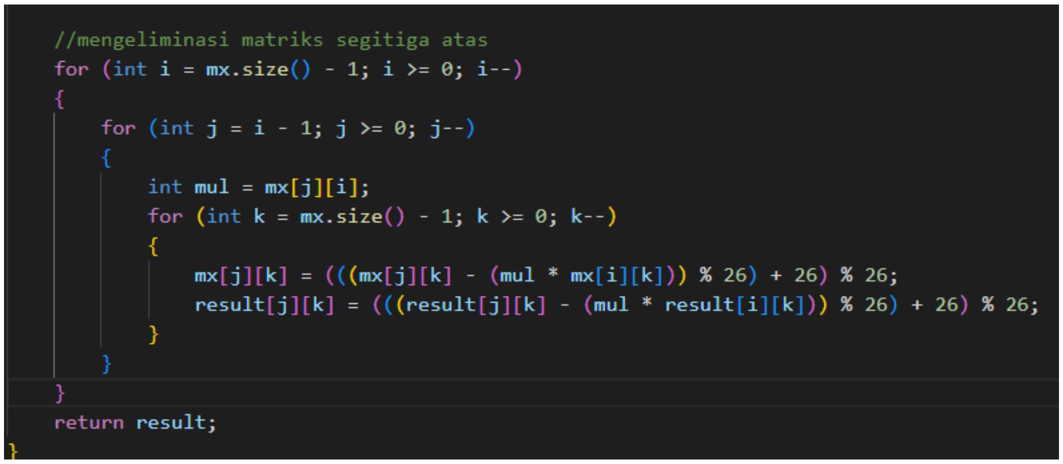


```cpp
string hillCipherDec(string cText, vector<vector<int>> key)
{
    //error handling
    if (cText.length() % key.size() != 0)
    {
        return "";
    }

    vector<vector<int>> mxPText, mxCText, invKey;
    string result = "";
    //konversi string ke matriks
    for (int i = 0; i < key.size(); i++)···

    //mencari inverse key dengan metode eliminasi gauss
    invKey = GaussianInvMod(key, 26);

    //perkalian inverse key dengan cipher text
    mxPText = MMultmod(invKey, mxCText, 26);

    //konversi matriks ke string
    for (int i = 0; i < mxPText[0].size(); i++)···
    return result;
}
```

Fungsi inverse matriks eliminasi gauss

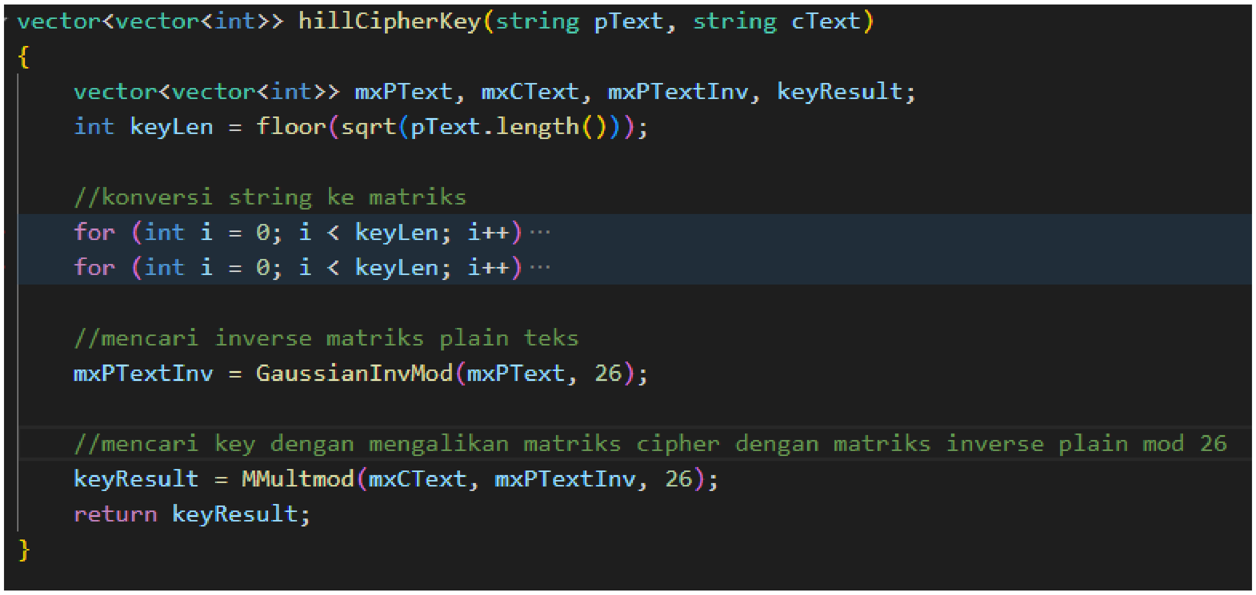


```
vector<vector<int>> GaussianInvMod(vector<vector<int>> mx, int mod)
{
    //matriks identitas untuk hasil matriks akhir
    vector<vector<int>> result = mxIdentity(mx.size());

    //error handling
    if (mx.size() != mx[0].size())···

    for (int i = 0; i < mx.size(); i++)
    {
        //error handilng jika elemen diagonal tidak koprima dengan 26
        if (InvMod(mx[i][i], 26) == 0)···
        int inverse = InvMod(mx[i][i], mod);

        //mengalikan elemen diagonal dengan inverse modulonya
        // 19   3   -> dikali 11 mod 26   -> (19*11) mod 26 = 1    (3*11) mod 26 = 7
        // 5    7
        // menjadi
        // 1    7
        // 5    7
        for (int j = 0; j < mx[i].size(); j++)
        {
            mx[i][j] = (mx[i][j] * inverse) % 26;
            result[i][j] = (result[i][j] * inverse) % 26;
        }

        //eliminasi gauss untuk baris selanjutnya
        // 1    7
        // 5    7   -> dikurangi R1 * 5
        for (int j = i + 1; j < mx.size(); j++)
        {
            int mul = mx[j][i];
            for (int k = 0; k < mx.size(); k++)
            {
                mx[j][k] = (((mx[j][k] - (mul * mx[i][k])) % 26) + 26) % 26;
                result[j][k] = (((result[j][k] - (mul * result[i][k])) % 26) + 26) % 26;
            }
        }
    }
    //hasil akhir merupakan matriks segitiga atas dengan semua elemen diagonal bernilai 1
```



```
    //mengeliminasi matriks segitiga atas
    for (int i = mx.size() - 1; i >= 0; i--)
    {
        for (int j = i - 1; j >= 0; j--)
        {
            int mul = mx[j][i];
            for (int k = mx.size() - 1; k >= 0; k--)
            {
                mx[j][k] = (((mx[j][k] - (mul * mx[i][k])) % 26) + 26) % 26;
                result[j][k] = (((result[j][k] - (mul * result[i][k])) % 26) + 26) % 26;
            }
        }
    }
    return result;
}
```

Fungsi hill cipher key



```cpp
vector<vector<int>> hillCipherKey(string pText, string cText)
{
    vector<vector<int>> mxPText, mxCText, mxPTextInv, keyResult;
    int keyLen = floor(sqrt(pText.length()));

    //konversi string ke matriks
    for (int i = 0; i < keyLen; i++)···
    for (int i = 0; i < keyLen; i++)···

    //mencari inverse matriks plain teks
    mxPTextInv = GaussianInvMod(mxPText, 26);

    //mencari key dengan mengalikan matriks cipher dengan matriks inverse plain mod 26
    keyResult = MMultmod(mxCText, mxPTextInv, 26);
    return keyResult;
}
```