Nama : Alvaro Dwi Oktaviano
NPM : 140810200041
Kelas : A

Tugas

1. Menjelaskan program hill-cipher yang sudah dibuat

Jawaban

1.

Kode hill cipher ini dibuat dengan C++

```
using matrix = std::vector<std::vector<int>>;
```

Untuk mempermudah operasi matriks didalamnya, dibuatlah sebuah alias matrix yang merupakan vector 2d

```
void getCofactor(matrix &, matrix &, int, int);
int determinant(matrix &, int);
void adjoint(matrix &, matrix &);
void displayMatrix(matrix);
```

Selain itu, dibuat juga fungsi fungsi matriks yang akan dipakai selama pengerjaan hill cipher, yaitu pencarian determinan, kofaktor, adjoint dan tentunya untuk menampilkan matriks. (note. Semuanya copas dari tutorialspoint)

```
while (text.length() != 0) {
    for (int i = 0; i < m; i++) {
      if (isupper(text[i]))
        msg[i][0] = text[i] - 65;
      else
        msg[i][0] = text[i] - 97;
    }

    for (int i = 0; i < m; i++) {
      for (int j = 0; j < 1; j++) {
        hasil[i][j] = 0;
        for (int k = 0; k < m; k++) {
          hasil[i][j] += key[i][k] * msg[k][j];
        }
        hasil[i][j] = hasil[i][j] % 26;
      }
    }

    for (int i = 0; i < m; i++) {
      if (isupper(text[i]))
        encryp += hasil[i][0] + 65;
      else
        encryp += hasil[i][0] + 97;
    }
    text.erase(0, m);
  }
```

Untuk **enkripsi**, setelah key dan plaintext diinput dan dipecah ke dalam submatriks, program tinggal melakukan perkalian matriks dan setelah itu hasilnya di kembalikan ke kode ascii.

Fungsi text.erase disini adalah modifier dari while loop yang menggunakan panjang teks untuk syarat nya

```
matrix adj(key.size(), std::vector<int>(key.size()));

  int det = determinant(key, m);
  adjoint(key, adj);

  int det_inv = 0;
  int flag = 0;

  for (int i = 0; i < 26; i++) {
    flag = mod((det * i), 26);

    if (flag == 1) {
      det_inv = i;
    }
  }

  matrix i_mtx(key.size(), std::vector<int>(key.size()));

  for (int i = 0; i < m; i++) {
    for (int j = 0; j < m; j++) {
      i_mtx[i][j] = mod((adj[i][j] * det_inv), 26);
    }
  }
```

Untuk **dekripsi**, program akan lebih dahulu mencari invers dari matriks key, yang menggunakan fungsi kofaktor, adjoin dan determinan matriks.

```
while (text.length() != 0) {
    for (int i = 0; i < m; i++) {
      if (isupper(text[i]))
        msg[i][0] = text[i] - 65;
      else
        msg[i][0] = text[i] - 97;
    }

    for (int i = 0; i < m; i++) {
      for (int j = 0; j < 1; j++) {
        hasil[i][j] = 0;
        for (int k = 0; k < m; k++) {
          hasil[i][j] += i_mtx[i][k] * msg[k][j];
        }
        hasil[i][j] = hasil[i][j] % 26;
      }
    }
    for (int i = 0; i < m; i++) {
      if (isupper(text[i]))
        decryp += hasil[i][0] + 65;
      else
        decryp += hasil[i][0] + 97;
    }
    text.erase(0, m);
  }
```

Setelah itu, program menjalankan operasi yang sama pada enkripsi, dengan perbedaan pada key yang dipakai.

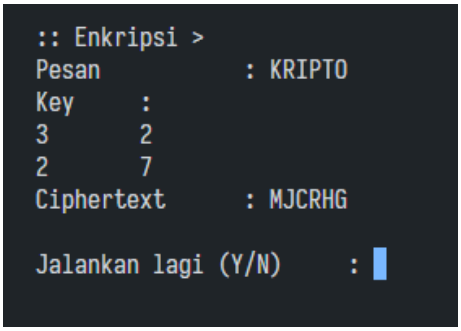
```
int k = 0;
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < m; j++) {
      if (isupper(plain[k])) {
        p[j][i] = plain[k] - 65;
      } else {
        p[j][i] = plain[k] - 97;
      }
      k++;
    }
  }

  k = 0;
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < m; j++) {
      if (isupper(cipher[k])) {
        c[j][i] = cipher[k] - 65;
      } else {
        c[j][i] = cipher[k] - 97;
      }
      k++;
    }
  }
```

Untuk **pencarian key** dari plaintext dan ciphertext, program akan mengelompokkan cipher text dan plaintext ke dalam matriks berdasarkan ordo layaknya pada enkripsi.

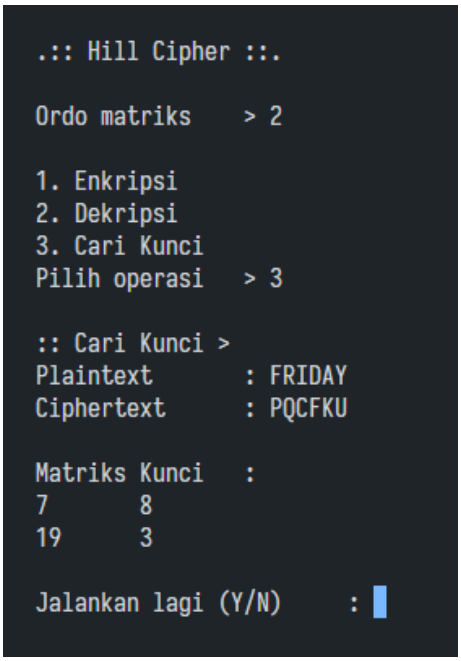
Setelah itu program akan mencari inverse dari matriks plaintext yang kemudian dikalikan terhadap matriks cipher text.

```
int det = determinant(p, m);
  adjoint(p, adj);

  int det_inv = 0;
  int flag = 0;

  for (int i = 0; i < 26; i++) {
    flag = mod((det * i), 26);

    if (flag == 1) {
      det_inv = i;
    }
  }

  for (int i = 0; i < m; i++) {
    for (int j = 0; j < m; j++) {
      p_inv[i][j] = mod((adj[i][j] * det_inv), 26);
    }
  }

  for (int i = 0; i < c.size(); i++) {
    for (int j = 0; j < p_inv.size(); j++) {
      for (int k = 0; k < c.size(); k++) {
        key[i][j] = mod((key[i][j] + c[i][k] * p_inv[k][j]), 26);
      }
    }
  }
```

Screenshot


:: Enkripsi >
Pesan : KRIPTO
Key :
3 2
2 7
Ciphertext : MJCRHG
Jalankan lagi (Y/N) :


:: Dekripsi >
Pesan : MJCRHG
Key Asli :
3 2
2 7
Invers dari Key :
5 6
6 17
Plaintext : KRIPTO
Jalankan lagi (Y/N) :


.:: Hill Cipher ::.
Ordo matriks > 2
1. Enkripsi
2. Dekripsi
3. Cari Kunci
Pilih operasi > 3
:: Cari Kunci >
Plaintext : FRIDAY
Ciphertext : PQCFKU
Matriks Kunci :
7 8
19 3
Jalankan lagi (Y/N) :