

NAMA : Nabil Ulil Albab
NPM : 23670010

The screenshot shows a terminal window with several tabs open. The active tab displays Go code for two encryption functions: `encryptCaesar` and `encryptRailFence`. The `encryptCaesar` function takes a plaintext string and a key integer, returning an encrypted string. It uses a simple substitution cipher where each letter is shifted by the key number of positions in the alphabet. The `encryptRailFence` function takes a plaintext string and a key integer, returning an encrypted string. It uses a rail fence cipher where the text is written in a zigzag pattern across multiple rails and then read off vertically. Both functions use the `strings.Builder` type to efficiently build the resulting strings.

On the right side of the terminal, there is a vertical status bar with various system information:

- User: kos
- hostname: fea
- Uptime: 4 s
- Distro: Fe4
- Kernel: Li4
- WM: Hy
- TERM: rvm
- CPU: i2z
- Disk: 73s
- Memory: 4.0

Below the status bar, there is a color palette labeled "colors".

The terminal also shows a menu-based program for encryption:

```
> go run main.go
===== MENU KRIPTOGRAFI =====
1. Enkripsi Caesar Cipher
2. Enkripsi Rail Fence Cipher
3. Keluar
Pilih menu (1-3): 1
Masukkan teks: NABILULILALBAB
Masukkan kunci Caesar (angka): 3
Hasil Caesar Cipher: QDELOXOLODOEDE
Tekan Enter untuk kembali ke menu...
===== MENU KRIPTOGRAFI =====
1. Enkripsi Caesar Cipher
2. Enkripsi Rail Fence Cipher
3. Keluar
Pilih menu (1-3): 2
Masukkan teks: NABILULILALBAB
Masukkan jumlah rails: 3
Hasil Rail Fence Cipher: NLLAAIUIABBBBLL
Tekan Enter untuk kembali ke menu...
===== MENU KRIPTOGRAFI =====
1. Enkripsi Caesar Cipher
2. Enkripsi Rail Fence Cipher
3. Keluar
Pilih menu (1-3):
```

At the bottom of the terminal, there is a status bar with the following information:

1 LSP ~ gopls uts-siber 25/1

```
package main

import (
    "fmt"
    "strings"
)

func encryptCaesar(plaintext string, key int) string {
    var ciphertext strings.Builder
    for _, char := range plaintext {
        if char >= 'A' && char <= 'Z' {
            newChar := 'A' + (char-'A'+rune(key))%26
            ciphertext.WriteRune(newChar)
        } else {
            ciphertext.WriteRune(char)
        }
    }
    return ciphertext.String()
```

```

}

func encryptRailFence(plaintext string, key int) string {
    rails := make([][]rune, key)
    railNum := 0
    direction := 1

    for _, char := range plaintext {
        rails[railNum] = append(rails[railNum], char)
        if railNum == 0 {
            direction = 1
        } else if railNum == key-1 {
            direction = -1
        }
        railNum += direction
    }

    var ciphertext strings.Builder
    for _, rail := range rails {
        for _, char := range rail {
            ciphertext.WriteRune(char)
        }
    }
    return ciphertext.String()
}

func pause() {
    var dummy string
    fmt.Println("\nTekan Enter untuk kembali ke menu...")
    fmt.Scanln(&dummy)
}

func main() {
    for {
        fmt.Println("\n===== MENU KRIPTOGRAFI =====")
        fmt.Println("1. Enkripsi Caesar Cipher")
        fmt.Println("2. Enkripsi Rail Fence Cipher")
        fmt.Println("3. Keluar")
        fmt.Print("Pilih menu (1-3): ")

        var pilihan int
        fmt.Scanln(&pilihan)

        if pilihan == 3 {
            fmt.Println("Terima kasih, program selesai.")
            break
        }

        var teks string
    }
}

```

```
fmt.Println("Masukkan teks: ")
fmt.Scanln(&teks)
plaintext := strings.ToUpper(teks)

switch pilihan {
case 1:
    var key int
    fmt.Println("Masukkan kunci Caesar (angka): ")
    fmt.Scanln(&key)
    cipher := encryptCaesar(plaintext, key)
    fmt.Printf("\nHasil Caesar Cipher: %s\n", cipher)
    pause()

case 2:
    var key int
    fmt.Println("Masukkan jumlah rails: ")
    fmt.Scanln(&key)
    cipher := encryptRailFence(plaintext, key)
    fmt.Printf("\nHasil Rail Fence Cipher: %s\n", cipher)
    pause()

default:
    fmt.Println("Pilihan tidak valid.")
    pause()
}

}
```