

LAPORAN PRAKTIKUM

Algoritma Pemrograman

MODUL 10

ELSE-IF



Disusun Oleh:

MUHAMAD RAFI ALFIANSYAH

109082500191

S1IF-13-04

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

LATIHAN KELAS – GUIDED

1. Guided 1

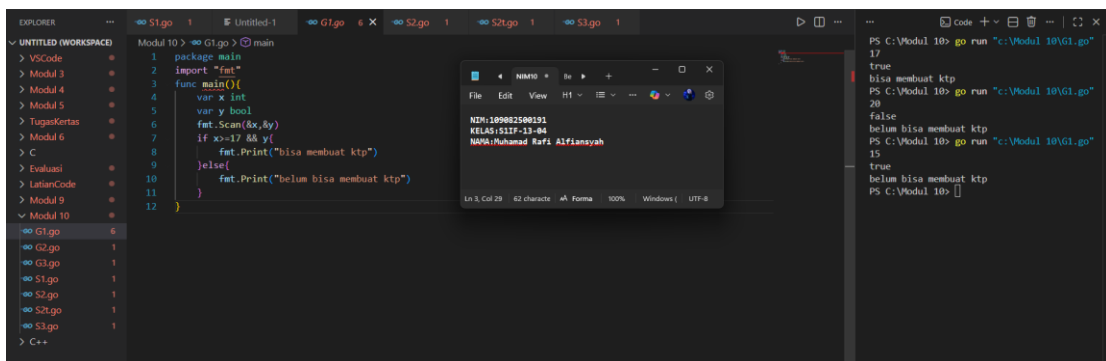
Source Code

```
package main

import "fmt"

func main(){
    var x int
    var y bool
    fmt.Scan(&x, &y)
    if x>=17 && y{
        fmt.Print("bisa membuat ktp")
    }else{
        fmt.Print("belum bisa membuat ktp")
    }
}
```

Screenshoot Program



Deskripsi Program

Program ini dibuat menggunakan bahasa pemrograman Go untuk menentukan apakah seseorang memenuhi syarat dalam pembuatan KTP. Penilaian dilakukan berdasarkan dua input, yaitu umur pengguna dan sebuah nilai boolean yang mewakili status tambahan tertentu. Program menerima input berupa bilangan bulat untuk umur dan nilai boolean (true atau false) untuk status tersebut.

Pada awal program, paket `fmt` diimpor untuk memungkinkan penggunaan fungsi input dan output. Di dalam fungsi `main`, dua variabel dideklarasikan: variabel `x` bertipe `int` untuk menyimpan umur, dan variabel `y` bertipe `bool` untuk menyimpan nilai

status boolean. Program kemudian membaca kedua nilai tersebut melalui fungsi `fmt.Scan(&x, &y)`.

Setelah input diterima, program melakukan proses pengambilan keputusan menggunakan struktur percabangan `if-else`. Kondisi yang diperiksa adalah apakah umur (x) bernilai 17 tahun atau lebih, dan apakah nilai boolean (y) bernilai `true`. Kedua kondisi ini dihubungkan menggunakan operator logika `&&` (AND). Jika kedua syarat terpenuhi, program mencetak pesan “bisa membuat ktp” yang berarti pengguna sudah memenuhi kriteria pembuatan KTP. Jika salah satu atau kedua syarat tidak terpenuhi, program akan menampilkan pesan “belum bisa membuat ktp”.

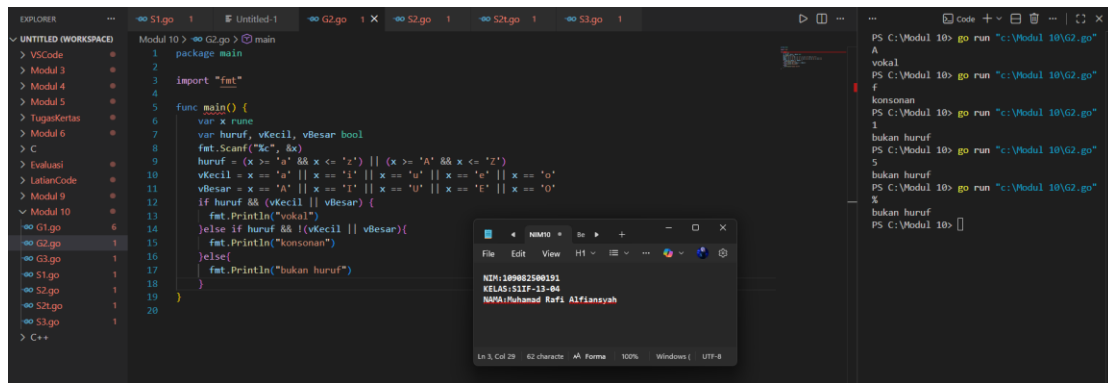
Secara keseluruhan, program ini menunjukkan penggunaan tipe data dasar, pembacaan input, serta pemanfaatan operator logika dalam pengambilan keputusan untuk menentukan kelayakan pembuatan KTP.

2. Guided 2

Source Code

```
package main
import "fmt"
func main() {
    var x rune
    var huruf, vKecil, vBesar bool
    fmt.Scanf("%c", &x)
    huruf = (x >= 'a' && x <= 'z') || (x >= 'A' && x <= 'Z')
    vKecil = x == 'a' || x == 'i' || x == 'u' || x == 'e' ||
x == 'o'
    vBesar = x == 'A' || x == 'I' || x == 'U' || x == 'E' ||
x == 'O'
    if huruf && (vKecil || vBesar) {
        fmt.Println("vokal")
    } else if huruf && !(vKecil || vBesar) {
        fmt.Println("konsonan")
    } else {
        fmt.Println("bukan huruf")
    }
}
```

Screenshoot Program



Deskripsi Program

Program ini berfungsi untuk menentukan apakah suatu karakter yang diinput oleh pengguna termasuk huruf vokal, huruf konsonan, atau bukan huruf. Proses penentuan dilakukan melalui pengecekan karakter berdasarkan nilai Unicode-nya, serta pemanfaatan variabel-variabel boolean yang mewakili kategori huruf tertentu.

Pertama, program mendeklarasikan sebuah variabel `x` bertipe `rune`. Tipe data `rune` digunakan karena dapat menyimpan nilai Unicode dari satu karakter, sehingga karakter yang dimasukkan dapat diperlakukan sebagai bilangan bulat untuk keperluan perbandingan. Selain itu, tiga variabel boolean dideklarasikan: `huruf`, `vKecil`, dan `vBesar`. Ketiga variabel ini berfungsi untuk menampung hasil evaluasi kondisi tertentu terkait kategori karakter.

Selanjutnya, program membaca satu karakter input menggunakan `fmt.Scanf("%c", &x)`. Hasil input tersebut disimpan dalam variabel `x`. Kemudian program menentukan apakah karakter tersebut merupakan huruf alfabet dengan memeriksa apakah `x` berada dalam rentang huruf kecil 'a' hingga 'z' atau huruf besar 'A' hingga 'Z'. Hasil pemeriksaan ini disimpan dalam variabel `huruf`.

Setelah menentukan apakah karakter merupakan huruf, program melanjutkan dengan mengidentifikasi apakah karakter tersebut merupakan huruf vokal. Pemeriksaan dilakukan dalam dua bagian, yaitu untuk huruf vokal kecil dan huruf vokal besar. Variabel `vKecil` bernilai `true` apabila karakter yang dimasukkan adalah salah satu dari 'a', 'i', 'u', 'e', atau 'o'. Begitu pula, `vBesar` bernilai `true` jika karakter yang dimasukkan adalah 'A', 'I', 'U', 'E', atau 'O'.

Hasil dari kedua variabel tersebut digabungkan pada percabangan utama. Jika suatu karakter terdeteksi sebagai huruf dan termasuk salah satu huruf vokal (baik huruf kecil maupun besar), program mencetak “vokal”. Jika karakter merupakan huruf tetapi bukan vokal, maka program mencetak “konsonan”. Apabila karakter tidak memenuhi syarat sebagai huruf alfabet, program menampilkan keluaran “bukan huruf”.

Secara keseluruhan, program ini memanfaatkan kombinasi tipe data rune, pemeriksaan rentang Unicode, dan variabel boolean untuk mengklasifikasikan input secara akurat menjadi vokal, konsonan, atau bukan huruf. Pendekatan ini memastikan pengecekan karakter dapat dilakukan secara efisien tanpa perlu mengubah format huruf terlebih dahulu.

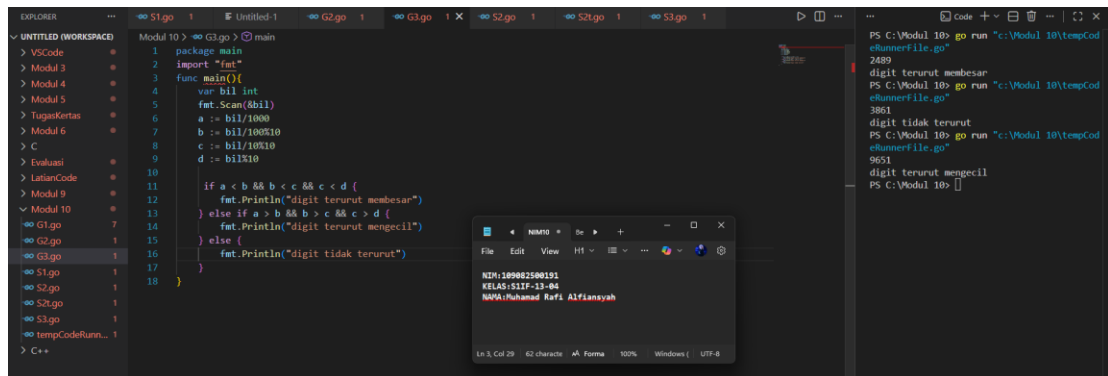
3. Guided 3

Source Code

```
package main
import "fmt"
func main(){
    var bil int
    fmt.Scan(&bil)
    a := bil/1000
    b := bil/100%10
    c := bil/10%10
    d := bil%10

    if a < b && b < c && c < d {
        fmt.Println("digit terurut membesar")
    } else if a > b && b > c && c > d {
        fmt.Println("digit terurut mengecil")
    } else {
        fmt.Println("digit tidak terurut")
    }
}
```

Screenshoot Program



The screenshot shows a Go program in VS Code and its execution output in the terminal. The program is located in `Modul 10` and is named `G3.go`. It defines a `main` function that reads an integer `bil` from the command line using `fmt.Scan(&bil)`. The program then extracts the four digits of `bil` into variables `a`, `b`, `c`, and `d` using integer division and modulus operations. It then checks if the digits are in ascending order (`a < b && b < c && c < d`) or descending order (`a > b && b > c && c > d`). If ascending, it prints "digit terurut membesar". If descending, it prints "digit terurut mengecil". Otherwise, it prints "digit tidak terurut".

```
1 package main
2 import "fmt"
3 func main() {
4     var bil int
5     fmt.Scan(&bil)
6     a := bil/1000
7     b := bil/100%10
8     c := bil/10%10
9     d := bil%10
10
11     if a < b && b < c && c < d {
12         fmt.Println("digit terurut membesar")
13     } else if a > b && b > c && c > d {
14         fmt.Println("digit terurut mengecil")
15     } else {
16         fmt.Println("digit tidak terurut")
17     }
18 }
```

The terminal output shows the program being run with the input `2489`. The output is:

```
PS C:\Modul 10> go run "c:\Modul 10\tempCodeRunnerFile.go"
2489
digit terurut membesar
PS C:\Modul 10> go run "c:\Modul 10\tempCodeRunnerFile.go"
3861
digit tidak terurut
PS C:\Modul 10> go run "c:\Modul 10\tempCodeRunnerFile.go"
9651
digit terurut mengecil
PS C:\Modul 10>
```

Deskripsi Program

rogram ini digunakan untuk menentukan apakah empat digit dari sebuah bilangan bulat empat angka tersusun dalam urutan membesar, urutan mengecil, atau tidak memiliki urutan tertentu. Proses ini dilakukan dengan memecah angka input menjadi empat digit terpisah, lalu membandingkannya berdasarkan aturan yang telah ditentukan.

Program diawali dengan deklarasi variabel `bil` bertipe integer yang berfungsi untuk menyimpan bilangan empat digit yang diinput oleh pengguna. Setelah itu, program membaca input tersebut menggunakan `fmt.Scan(&bil)`.

Untuk memperoleh masing-masing digit, program melakukan operasi pembagian dan modulus. Digit pertama (ribuan) disimpan pada variabel `a` dengan operasi `bil / 1000`. Digit kedua (ratusan) dihitung melalui operasi `bil / 100 % 10`, sehingga hanya nilai ratusannya yang tersisa. Digit ketiga (puluhan) diperoleh melalui `bil / 10 % 10`. Digit terakhir (satuan) diperoleh dengan `bil % 10`. Dengan teknik ini, setiap digit dari bilangan dapat dipisahkan dan dianalisis secara terpisah.

Setelah semua digit berhasil dipisahkan, program melakukan pengecekan pola urutan digit menggunakan struktur percabangan. Kondisi pertama memeriksa apakah digit-digit tersebut tersusun membesar, yaitu apakah $a < b$, $b < c$, dan $c < d$. Jika semua kondisi bernilai benar, maka program menampilkan pesan "digit terurut membesar". Kondisi kedua memeriksa apakah digit-digit tersebut tersusun mengecil dengan aturan kebalikan, yaitu $a > b$, $b > c$, dan $c > d$. Jika kondisi ini terpenuhi, program mencetak "digit terurut mengecil".

Jika kedua kondisi tersebut tidak terpenuhi, berarti digit-digit tidak membentuk urutan membesar maupun mengecil. Dalam kasus tersebut, program menampilkan keluaran “digit tidak teratur”.

Secara keseluruhan, program ini menunjukkan bagaimana operasi pembagian dan modulus dapat digunakan untuk memecah bilangan menjadi digit-digit penyusunnya, serta bagaimana percabangan logika dimanfaatkan untuk menganalisis pola urutan data numerik.

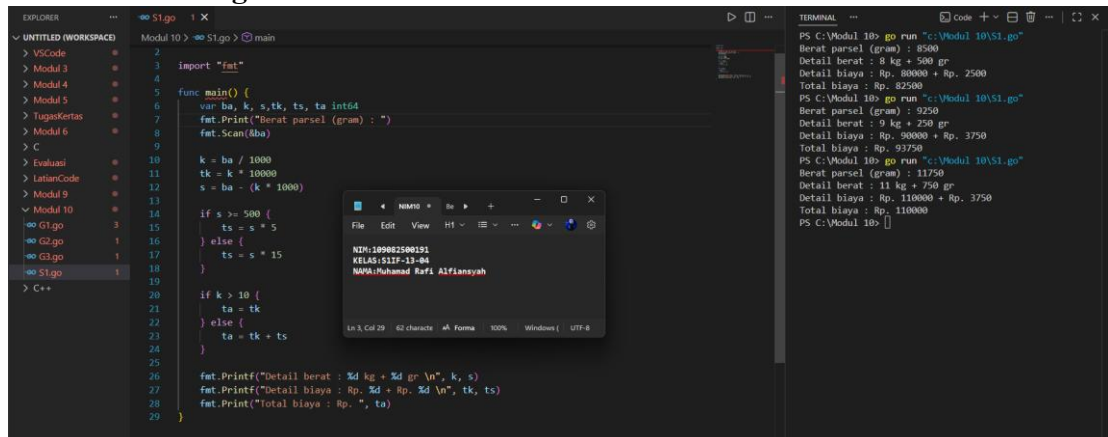
TUGAS

1. Tugas 1

Source Code

```
package main
import "fmt"
func main() {
    var ba, k, s, tk, ts, ta int64
    fmt.Print("Berat parsel (gram) : ")
    fmt.Scan(&ba)
    k = ba / 1000
    tk = k * 10000
    s = ba - (k * 1000)
    if s >= 500 {
        ts = s * 5
    } else {
        ts = s * 15
    }
    if k > 10 {
        ta = tk
    } else {
        ta = tk + ts
    }
    fmt.Printf("Detail berat : %d kg + %d gr \n", k, s)
    fmt.Printf("Detail biaya : Rp. %d + Rp. %d \n", tk, ts)
    fmt.Print("Total biaya : Rp. ", ta)
}
```

Screenshoot Program



```
Modul 10 > S1go > main
3 import "fmt"
4
5 func main() {
6     var ba, k, s, tk, ts, ta int64
7     fmt.Println("Berat parcel (gram) : ")
8     fmt.Scan(&ba)
9
10    k = ba / 1000
11    tk = k * 10000
12    s = ba - (k * 1000)
13
14    if s >= 500 {
15        ts = s * 5
16    } else {
17        ts = s * 15
18    }
19
20    if k > 10 {
21        ta = tk
22    } else {
23        ta = tk + ts
24    }
25
26    fmt.Printf("Detail berat : %d kg + %d gr \n", k, s)
27    fmt.Printf("Detail biaya : Rp. %d + Rp. %d \n", tk, ts)
28    fmt.Println("Total biaya : Rp. ", ta)
29 }
```

```
PS C:\Modul 10> go run ".\S1go.go"
Berat parcel (gram) : 8500
Detail berat : 8 kg + 500 gr
Detail biaya : Rp. 80000 + Rp. 2500
Total biaya : Rp. 82500
PS C:\Modul 10> go run ".\S1go.go"
Berat parcel (gram) : 9250
Detail berat : 9 kg + 250 gr
Detail biaya : Rp. 90000 + Rp. 3750
Total biaya : Rp. 93750
PS C:\Modul 10> go run ".\S1go.go"
Berat parcel (gram) : 11750
Detail berat : 11 kg + 750 gr
Detail biaya : Rp. 110000 + Rp. 3750
Total biaya : Rp. 110000
PS C:\Modul 10>
```

Deskripsi Program

Program ini berfungsi untuk menghitung biaya pengiriman parcel berdasarkan total berat yang dimasukkan pengguna. Berat parcel diolah dengan memisahnya menjadi kilogram dan gram, lalu biaya dihitung sesuai aturan tarif: setiap kilogram dikenakan Rp10.000, sedangkan sisa gram dikenakan tarif berbeda tergantung jumlah gram yang tersisa.

Program diawali dengan deklarasi variabel untuk menyimpan berat total, hasil konversi berat (kg dan gram), serta biaya berdasarkan kilogram, gram, dan total keseluruhan. Pengguna memasukkan berat parcel dalam gram, kemudian program membaginya menjadi berat kilogram ($ba / 1000$) dan sisa gram ($ba - k * 1000$).

Biaya kilogram dihitung dengan mengalikan jumlah kilogram dengan Rp10.000. Untuk sisa gram, jika jumlahnya 500 gram atau lebih, tarif yang digunakan adalah Rp5 per gram; jika kurang dari 500 gram, tarifnya Rp15 per gram. Selanjutnya, apabila berat parcel lebih dari 10 kilogram, biaya sisa gram tidak dihitung dan total biaya hanya berasal dari biaya kilogram. Jika tidak, total biaya merupakan penjumlahan biaya kilogram dan biaya gram.

Program kemudian menampilkan rincian berat, biaya masing-masing komponen, serta total biaya yang harus dibayar.

2. Tugas 2

Source Code

```
package main

import "fmt"

func main() {

    var nam float64

    var nmk string

    fmt.Print("Nilai akhir mata kuliah : ")

    fmt.Scan(&nam)

    if nam > 80 {

        nmk = "A"

    } else if nam > 72.5 {

        nmk = "AB"

    } else if nam > 65 {

        nmk = "B"

    } else if nam > 57.5 {

        nmk = "BC"

    } else if nam > 50 {

        nmk = "C"

    } else if nam > 40 {

        nmk = "D"

    } else {

        nmk = "E"

    }

    fmt.Print("Nilai mata kuliah : ", nmk)}
```

Screenshoot Program

The screenshot shows a Go program in VS Code and its execution output in a terminal window. The program is located in `Modul 10` and `S2.go`. It defines a `main` function that takes a float64 value `nam` and a string `nmk`. It prints the final grade and then scans the input value. The terminal output shows the program being run with the input `80.1`, resulting in the output `Nilai akhir mata kuliah : 80.1` and `Nilai mata kuliah : A`.

```
package main

import "fmt"

func main() {
    var nam float64
    var nmk string
    fmt.Print("Nilai akhir mata kuliah : ")
    fmt.Scan(&nam)
    if nam > 80 {
        nmk = "A"
    } else if nam > 72.5 {
        nmk = "AB"
    } else if nam > 65 {
        nmk = "B"
    } else if nam > 57.5 {
        nmk = "BC"
    } else if nam > 50 {
        nmk = "C"
    } else if nam > 40 {
        nmk = "D"
    } else {
        nmk = "E"
    }
    fmt.Print("Nilai mata kuliah : ", nmk)
}
```

```
PS C:\Modul 10> go run ".\S2.go"
Nilai akhir mata kuliah : 80.1
Nilai mata kuliah : A
PS C:\Modul 10>
```

a. Jika nam diberikan adalah 80.1, apa keluaran dari program tersebut?

Apakah eksekusi program tersebut sesuai spesifikasi soal?

Jika nilai yang dimasukkan adalah 80.1, program akan mengeksekusi kondisi `nam > 80`, sehingga menghasilkan output:

Nilai mata kuliah : A

Secara output, hasil ini benar. Namun, bila dibandingkan dengan spesifikasi nilai pada soal, logika program belum sepenuhnya sesuai karena batasan setiap rentang nilai tidak didefinisikan dengan jelas.

b. Apa saja kesalahan dari program tersebut? Mengapa demikian? Jelaskan alur program seharusnya!

1. Rentang nilai tidak tepat, sehingga beberapa nilai berada pada kategori yang salah (contoh: 70.6 seharusnya B, tetapi program memasukkannya ke AB).

2. Penggunaan tanda `>` tanpa batas bawah menyebabkan nilai tertentu tidak masuk kategori yang tepat (misal: nilai 80 tidak dianggap A).

3. Rentang kategori tidak konsisten dan tidak mengikuti standar penilaian yang diharapkan pada soal.

4. Tidak ada validasi nilai, seperti nilai di bawah 0 atau di atas 100.

Alur program seharusnya menetapkan rentang yang jelas dan tidak tumpang tindih, misalnya menggunakan `>=` pada batas bawah dan `<` pada batas atas.

c. Perbaiki program tersebut! Ujilah dengan masukan: 93.5; 70.6; dan 49.5. Seharusnya keluaran yang diperoleh adalah 'A', 'B', dan 'D'.

Source Code

```
package main

import "fmt"

func main() {

    var nam float64
    var nmk string

    fmt.Print("Nilai akhir mata kuliah : ")

    fmt.Scan(&nam)

    if nam > 80 {

        nmk = "A"

    } else if nam > 72.5 {

        nmk = "AB"

    } else if nam > 65 {

        nmk = "B"

    } else if nam > 57.5 {

        nmk = "BC"

    } else if nam > 50 {

        nmk = "C"

    } else if nam > 40 {

        nmk = "D"

    } else {

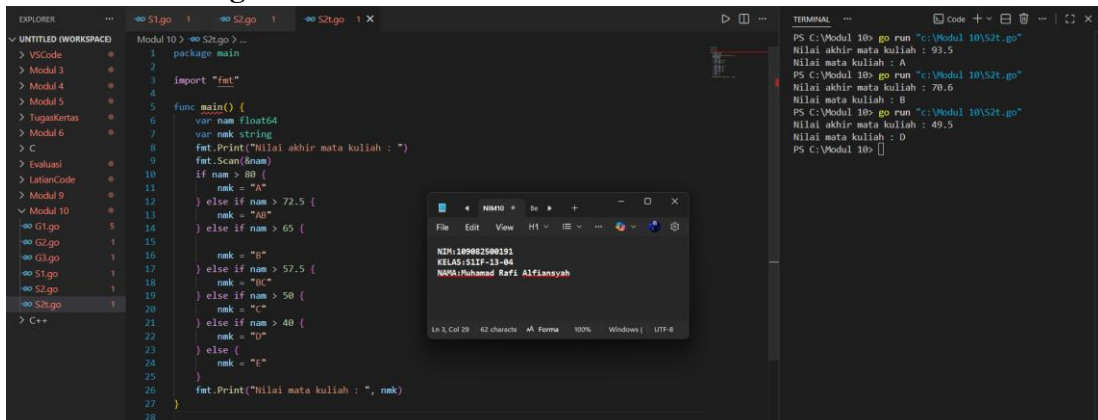
        nmk = "E"

    }

    fmt.Print("Nilai mata kuliah : ", nmk)

}
```

Screenshoot Program



Deskripsi Program

Program ini diawali dengan mendefinisikan paket main serta mengimpor paket `fmt` yang diperlukan untuk proses input dan output. Di dalam fungsi main, program mendeklarasikan dua variabel yaitu `nam` bertipe `float64` untuk menyimpan nilai akhir mata kuliah dalam bentuk angka desimal dan `nmk` bertipe `string` untuk menampung nilai huruf hasil konversi. Program kemudian meminta pengguna memasukkan nilai akhir melalui perintah `fmt.Print` dan membaca input menggunakan `fmt.Scan(&nam)`, yang menyimpan nilai tersebut ke variabel `nam`. Setelah nilai diterima, program menjalankan proses penentuan nilai huruf menggunakan serangkaian percabangan `if – else if – else`. Percabangan pertama memeriksa apakah nilai `nam` lebih besar dari 80; jika benar, nilai hurufnya adalah “A”. Jika tidak, program melanjutkan ke kondisi berikutnya untuk memeriksa apakah `nam` lebih besar dari 72.5 sehingga diberi nilai “AB”. Kondisi selanjutnya memeriksa nilai di atas 65 untuk kategori “B”, di atas 57.5 untuk kategori “BC”, di atas 50 untuk kategori “C”, dan di atas 40 untuk kategori “D”. Jika tidak ada satu pun kondisi yang terpenuhi, program menetapkan nilai “E” sebagai nilai huruf. Penggunaan percabangan berurutan ini memastikan bahwa setiap nilai hanya cocok dengan satu kategori karena program berhenti pada kondisi pertama yang bernilai benar. Setelah nilai huruf ditentukan, program menampilkan hasil akhir melalui perintah `fmt.Print("Nilai mata kuliah : ", nmk)` yang mencetak nilai huruf sesuai dengan kategori yang telah ditentukan berdasarkan input pengguna.

3. Tugas 3

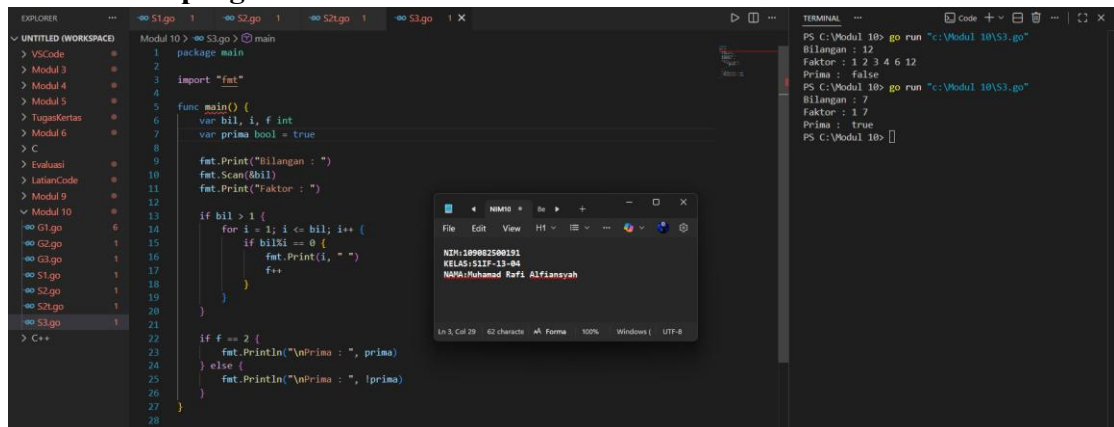
Source code

```
package main

import "fmt"

func main() {
    var bil, i, f int
    var prima bool = true
    fmt.Print("Bilangan : ")
    fmt.Scan(&bil)
    fmt.Print("Faktor : ")
    if bil > 1 {
        for i = 1; i <= bil; i++ {
            if bil%i == 0 {
                fmt.Print(i, " ")
                f++
            }
        }
    }
    if f == 2 {
        fmt.Println("\nPrima : ", prima)
    } else {
        fmt.Println("\nPrima : ", !prima)
    }
}
```

Screenshoot program



Deskripsi Program

Program ini digunakan untuk menampilkan faktor-faktor dari sebuah bilangan sekaligus menentukan apakah bilangan tersebut merupakan bilangan prima. Program diawali dengan deklarasi variabel `bil` sebagai bilangan yang akan diperiksa, variabel penghitung `i`, variabel `f` sebagai penghitung jumlah faktor, serta variabel boolean `prima` yang diinisialisasi dengan nilai `true`. Pengguna memasukkan sebuah bilangan melalui input yang disimpan ke variabel `bil`. Setelah itu program mencetak teks "Faktor :" sebagai penanda proses berikutnya. Jika bilangan yang dimasukkan lebih besar dari 1, program menjalankan perulangan `for` dari 1 hingga nilai bilangan tersebut. Pada setiap iterasi, program mengecek apakah `bil % i == 0`. Jika kondisi ini benar, berarti `i` merupakan faktor dari bilangan tersebut, sehingga nilai `i` dicetak dan variabel `f` bertambah satu untuk menghitung jumlah faktor yang ditemukan. Setelah perulangan selesai, program memeriksa apakah nilai `f` sama dengan 2. Jika jumlah faktor tepat dua, berarti bilangan tersebut adalah bilangan prima, sehingga program mencetak nilai `prima` yang bernilai `true`. Jika jumlah faktornya tidak sama dengan dua, program mencetak `!prima` yang bernilai `false`, menandakan bahwa bilangan tersebut bukan bilangan prima. Program ini menunjukkan cara kerja pemeriksaan faktor menggunakan operasi modulus, serta pemanfaatan perulangan dan logika boolean untuk menentukan keprimaannya secara sistematis.