

**LAPORAN PRAKTIKUM  
ALGORITMA PEMOGRAMAN 2**

**MODUL 11**

**MATERI**

**REVIEW STRUKTUR KONTROL**



Oleh:

**NAMA : FEBRIAN FALIH ALWAFI**

**NIM : 2311102181**

**KELAS : S1F-11-02**

**S1 TEKNIK INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

## I. DASAR TEORI

### Struktur Program Go

Dalam kerangka program yang ditulis dalam bahasa pemrograman Go, program utama selalu mempunyai dua komponen berikut:

- **package main** merupakan penanda bahwa file ini berisi program utama.
- **func main()** berisi kode utama dari sebuah program Go.

### Koding, Kompilasi, dan Eksekusi Go

#### ➤ Koding

- Tidak berbeda dengan penulisan program sumber dalam bahasa lain, program Go harus dibuat menggunakan penyunting teks dan disimpan dalam format teks.
- Setiap program go disimpan dalam file teks dengan ekstensi go, dengan nama bebas.
- Setiap satu program lengkap Go disimpan dalam satu folder tersendiri. Nama folder merupakan nama program tersebut.. Karena itu secara prinsip, satu program Co dapat dipecah dalam beberapa file dengan ekstensi go selama disimpan dalam folder yang sama.

#### ➤ Kompilasi

Beberapa bahasa pemrograman dirancang untuk diimplementasikan sebagai interpreter dan lainnya sebagai kompilator. Interpreter akan membaca setiap baris instruksi dan kemudian langsung mengeksekusinya, dengan hanya sedikit pemeriksaan apakah penulisan keseluruhan program sudah benar atau belum. Kompilator akan memeriksa keseluruhan program sumber dan kemudian mengubahnya menjadi program executable, sehingga konsistensi penulisan (seperti penggunaan tipe data) sudah diperiksa sebelum eksekusi.

Berikut adalah contoh sesi yang biasa dilakukan saat mengkompilasi dan mengeksekusi program dalam bahasa Go

- Panggil shell atau terminal (program/utiliti cmd.exe di Windows)
- Masuk ke dalam (cd) folder program (normalnya ada di C:\Users\golsrc) atau yang sejenis)
- Kemudian panggil perintah go build atau go build file go untuk mengkompilasi file.go
- Jika gagal, akan muncul pesan eror yang sesuai, pelajari dengan bairt pesan tersebut. perbaiki teks program sumber, kemudian ulangi proses build-nya.
- Jika berhasil maka pada folder tersebut akan dibuat program dengan nama yang sama dan diakhiri dengan .exe (untuk Windows)
- Panggil program eksekutabel tersebut dari terminal yang sama. Jangan memanggil program tersebut dengan mengklik eltselutabel tersebut dari folder harena program kalian hanya berbasis teks, bukan belum dirancang dengan tampilan Windows.

Semua proses terkait bahasa Go dilakukan melalui utilitas go. Beberapa opsi dengan utilitas go

- go build: mengkompilasi program sumber yang ada dalam folder menjadi sebuah program.
- go build file.go: mengkompilasi program sumber file.go saja.
- go fmt: membaca semua program sumber dalam folder dan mereformat penulisannya agar sesuai dengan standar penulisan program sumber Go.
- go clean: membersihkan file-file dalam folder sehingga tersisa program sumber nya saja.

## ➤ **TIPE DATA**

- **Tipe Data Nyata**

Tipe data riil digunakan untuk menyimpan angka pecahan. Go menawarkan dua opsi: float32 dan float64.

Contoh deklarasi variabel riil:

```
var x float32 = 3.14
var y float64 = 3.1415926535897
```

- **Tipe Data String**

String merupakan sebuah data yang berisi kumpulan 0 atau lebih karakter. Di Golang, tipe data string ditulis dengan kunci string. Untuk membuat nilai dengan tipe data string kita menuliskan kumpulan karakter yang dibungkus dengan petik dua (").

Contoh Program :

```
package main

import "fmt"

func main() {
    fmt.Println("Saya sedang belajar Go")
}
```

- **Tipe Data Boolean**

Boolean adalah tipe data yang hanya memiliki dua nilai yaitu benar atau salah. Tipe data boolean memang merupakan tipe data yang sangat sederhana.

Contoh Program :

```
package main

import "fmt"

func main() {
    fmt.Println(true) // bernilai benar
    fmt.Println(false) // bernilai salah
}
```

## II. GUIDED

- 1.) Telusuri program berikut dengan cara mengkompilasi dan mengeksekusi program. Silakan masukan data yang sesuai sebanyak yang diminta program. Perhatikan keluaran yang diperoleh. Coba terangkan apa sebenarnya yang dilakukan program tersebut?

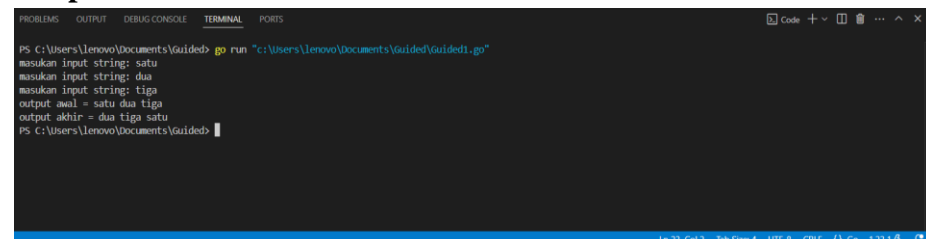
### Source Code :

```
package main

import "fmt"

func main() {
    var (
        satu, dua, tiga string
        temp      string
    )
    fmt.Print("masukan input string: ")
    fmt.Scanln(&satu)
    fmt.Print("masukan input string: ")
    fmt.Scanln(&dua)
    fmt.Print("masukan input string: ")
    fmt.Scanln(&tiga)
    fmt.Println("output awal = " + satu + " " + dua + " " +
tiga)
    temp = satu
    satu = dua
    dua = tiga
    tiga = temp
    fmt.Println("output akhir = " + satu + " " + dua + " " +
tiga)
}
```

### Output :



```
PS C:\Users\lenovo\Documents\Guided> go run "c:\Users\lenovo\Documents\Guided\Guided1.go"
masukan input string: satu
masukan input string: dua
masukan input string: tiga
output awal = satu dua tiga
output akhir = dua tiga satu
PS C:\Users\lenovo\Documents\Guided>
```

## Deskripsi Program :

Program diatas menampilkan 3 input string, string satu, dua, tiga. Setelah tiga string dimasukkan, program menampilkan urutan input seperti yang dimasukkan oleh pengguna: satu dua tiga. Program kemudian memodifikasi urutan string dengan memindahkan string pertama ke akhir. Dengan demikian, urutan baru menjadi: dua tiga satu. Program ini memindahkan elemen pertama dari input dan menambahkannya ke akhir daftar.

- 2.) Tahun kabisat adalah tahun yang habis dibagi 400 atau habis dibagi 4 tetapi tidak habis dibagi 100. Buatlah sebuah program yang menerima input sebuah bilangan bulat dan memeriksa apakah bilangan tersebut merupakan tahun kabisat (true) atau bukan (false).

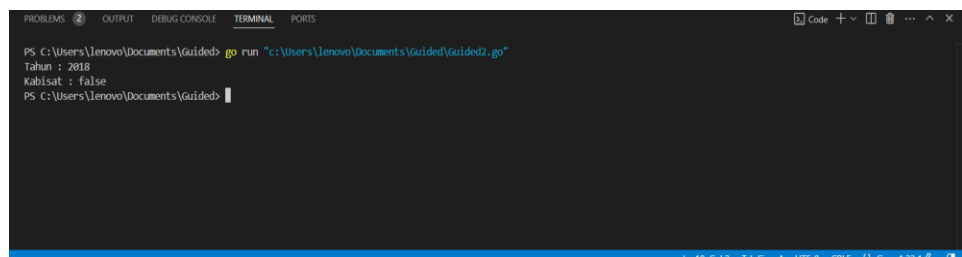
## Source Code :

```
package main

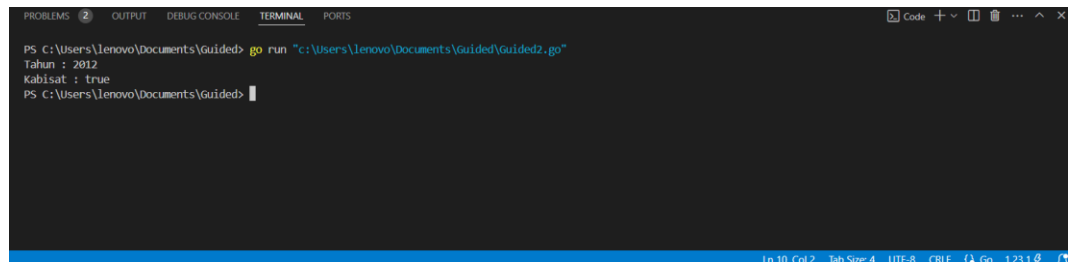
import "fmt"

func main() {
    var tahun int
    fmt.Print("Tahun : ")
    fmt.Scan(&tahun)
    fmt.Printf("Kabisat : %t \n", (tahun%4 == 0 && (tahun%100 != 0 ||
tahun%400 == 0)))
}
```

## Output :



```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\lenovo\Documents\Guided> go run "c:\Users\lenovo\Documents\Guided\Guided2.go"
Tahun : 2018
Kabisat : false
PS C:\Users\lenovo\Documents\Guided> |
```



```
PS C:\Users\lenovo\Documents\Guided> go run "c:\Users\lenovo\Documents\Guided\Guided2.go"
Tahun : 2012
Kabisat : true
PS C:\Users\lenovo\Documents\Guided> |
```

## Deskripsi Program :

Program tersebut meminta pengguna diminta untuk memasukkan sebuah tahun. Dalam contoh ini, input yang diberikan adalah tahun 2018. Setelah memproses logika tersebut, program memberikan output berupa yang kabiset false yang berarti tahun 2018 bukan tahun kabiset. Program ini menggunakan logika berdasarkan pembagian dan menampilkan hasilnya dalam bentuk teks di layar.

3.) Buat program Bola yang menerima input jari-jari suatu bola (bilangan bulat). Tampilkan Volume dan Luas kulit bola.  $volumebola = \frac{4}{3}\pi r^3$  dan  $luasbola = 4\pi r^2$  ( $\pi \approx 3.1415926535$ ).

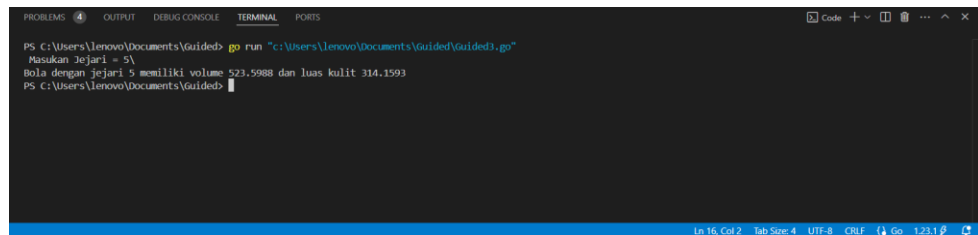
## Source Code :

```
package main

import (
    "fmt"
    "math"
)

func main() {
    var jari, volume, luasKulit float64
    fmt.Print(" Masukan Jejari = ")
    fmt.Scan(&jari)
    volume = math.Pi * (4.0 / 3.0) * math.Pow(jari, 3)
    luasKulit = 4 * math.Pi * math.Pow(jari, 2)
    fmt.Printf("Bola dengan jejari %v memiliki volume %.4f dan luas kulit %.4f\n", jari, volume, luasKulit)
}
```

## Output :



```
PS C:\Users\lenovo\Documents\Guided> go run "c:\Users\lenovo\Documents\Guided\Guided3.go"
Bola dengan jejari 5 memiliki volume 523.5988 dan luas kulit 314.1593
PS C:\Users\lenovo\Documents\Guided>
```

### Deskripsi Program :

Program di atas menghitung **volume** dan **luas permukaan** sebuah bola berdasarkan nilai jejari (radius) yang dimasukkan oleh pengguna. Program ini ditulis menggunakan bahasa Go dan menghasilkan perhitungan berdasarkan rumus geometri untuk bola. Program menginputkan jejari bola 5 dan menghasilkan volume bola 523.5988 dan luas permukaan bola = 314.1593. Program ini menggunakan rumus matematika dasar untuk menghitung volume dan luas permukaan bola.

## III. UNGUIDED

### 2b\_1.) Source Code :

```
package main

import "fmt"

func main() {
    // Array dengan urutan warna yang benar
    warnaTrue := [4]string{"merah", "kuning", "hijau", "ungu"}
    benar := true

    // Loop untuk 5 percobaan
    for i := 1; i <= 5; i++ {
        var inputWarna [4]string
        fmt.Printf("Percobaan %d: ", i)

        // Mengambil input untuk 4 warna
        fmt.Scan(&inputWarna[0], &inputWarna[1], &inputWarna[2],
        &inputWarna[3])

        // Mengecek setiap input warna apakah sesuai dengan urutan yang
        benar
        for j := 0; j < len(warnaTrue); j++ {
            if inputWarna[j] != warnaTrue[j] {
                benar = false
                break // Jika salah, langsung tandai salah dan lanjut ke
                percobaan berikutnya
            }
        }
    }
}
```



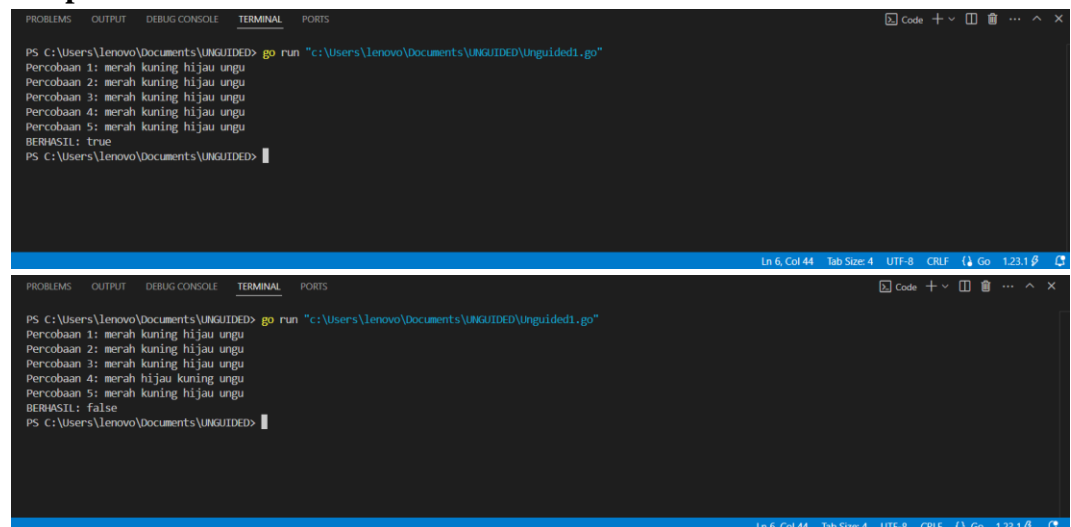
```

    }
  }
}

// Menampilkan hasil akhir setelah 5 percobaan
if benar {
    fmt.Println("BERHASIL: true")
} else {
    fmt.Println("BERHASIL: false")
}
}

```

### Output :



```

PS C:\Users\lenovo\Documents\UNGUIDED> go run "c:\Users\lenovo\Documents\UNGUIDED\unguided1.go"
Percobaan 1: merah kuning hijau ungu
Percobaan 2: merah kuning hijau ungu
Percobaan 3: merah kuning hijau ungu
Percobaan 4: merah kuning hijau ungu
Percobaan 5: merah kuning hijau ungu
BERHASIL: true
PS C:\Users\lenovo\Documents\UNGUIDED>

PS C:\Users\lenovo\Documents\UNGUIDED> go run "c:\Users\lenovo\Documents\UNGUIDED\unguided1.go"
Percobaan 1: merah kuning hijau ungu
Percobaan 2: merah kuning hijau ungu
Percobaan 3: merah kuning hijau ungu
Percobaan 4: merah hijau kuning ungu
Percobaan 5: merah kuning hijau ungu
BERHASIL: false
PS C:\Users\lenovo\Documents\UNGUIDED>

```

### Deskripsi Program :

Program yang meminta pengguna untuk memasukkan urutan warna sebanyak 5 kali percobaan. Program tersebut kemudian memeriksa apakah urutan warna yang dimasukkan setiap kali sesuai dengan urutan yang benar, yaitu merah, kuning, hijau, dan ungu. Setelah 5 percobaan, program menampilkan salah maka program akan menampilkan false yang berarti bahwa semua percobaan gagal, tapi jika **"BERHASIL: true"** yang berarti bahwa semua percobaan sukses, dan urutan warna yang dimasukkan pengguna benar sesuai dengan yang diharapkan.

### 2b\_2.) Source Code :

```

package main

import "fmt"

func main() {
    var pita string
    var banyak int

    fmt.Print("Masukkan jumlah (N) : ")
    fmt.Scan(&banyak)

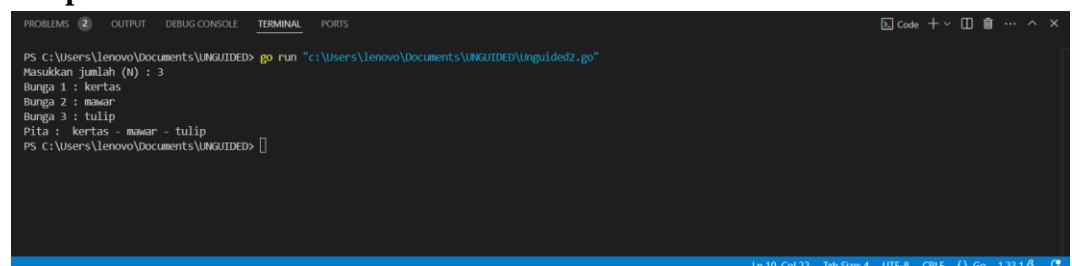
    // Loop untuk meminta input bunga sebanyak jumlah yang ditentukan
    for i := 0; i < banyak; i++ {
        var bunga string
        fmt.Printf("Bunga %d : ", i+1)
        fmt.Scan(&bunga)

        // Menambahkan nama bunga ke pita dengan pemisah " - "
        if i == banyak-1 {
            pita += bunga // Menghindari tanda " - " di akhir
        } else {
            pita += bunga + " - "
        }
    }

    // Menampilkan hasil
    fmt.Println("Pita : ", pita)
}

```

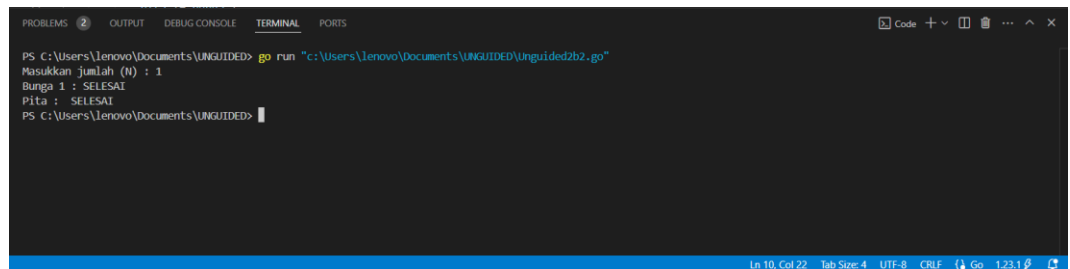
## Output :



```

PS C:\Users\lenovo\Documents\UNGUIDED> go run "c:\Users\lenovo\Documents\UNGUIDED\unguided2.go"
Masukkan jumlah (N) : 3
Bunga 1 : kertas
Bunga 2 : mawar
Bunga 3 : tulip
Pita : kertas - mawar - tulip
PS C:\Users\lenovo\Documents\UNGUIDED>

```



```
PS C:\Users\lenovo\Documents\UNGUIDED> go run "c:\Users\lenovo\Documents\UNGUIDED\Unguided2b2.go"
Masukkan jumlah (N) : 1
Bunga 1 : SELESAI
Pita : SELESAI
PS C:\Users\lenovo\Documents\UNGUIDED>
```

## Deskripsi Program :

Program ini meminta pengguna untuk memasukkan nama bunga sebanyak N kali, di mana N adalah jumlah yang dimasukkan oleh pengguna. Setelah pengguna memasukkan semua nama bunga, program akan menggabungkannya menjadi satu string yang dipisahkan oleh " - ". Akhirnya, program mencetak hasilnya dan program ini memberikan fleksibilitas kepada pengguna untuk memasukkan jumlah bunga yang diinginkan dan menampilkan hasilnya dengan cara yang rapi.

## 2b\_3.) Source Code :

```
package main

import "fmt"

func main() {
    var beratKiri, beratKanan float64

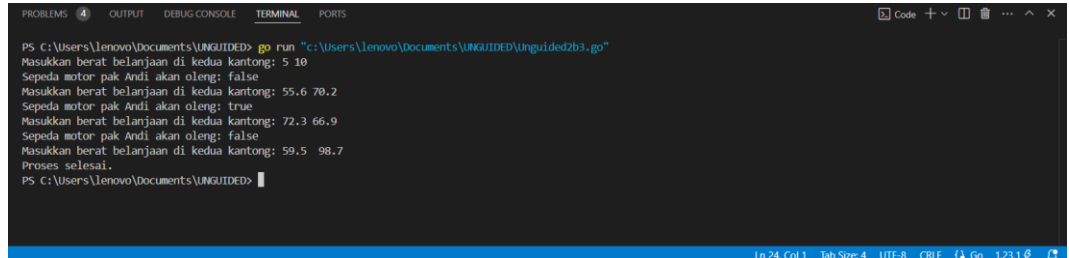
    for {
        fmt.Print("Masukkan berat belanjaan di kedua kantong: ")
        fmt.Scan(&beratKiri, &beratKanan)

        // Cek apakah total berat melebihi batas atau ada berat negatif
        if beratKiri < 0 || beratKanan < 0 || (beratKiri+beratKanan) > 150 {
            break
        }

        // Hitung apakah sepeda motor akan oleng
        sepedaMotorOleng := (beratKiri - beratKanan) >= 9 || (beratKanan
- beratKiri) >= 9
        fmt.Printf("Sepeda motor pak Andi akan oleng: %v\n",
sepedaMotorOleng)
    }

    fmt.Println("Proses selesai.")
}
```

## Output :



```
PS C:\Users\lenovo\Documents\UNGUIDED> go run "c:\Users\lenovo\Documents\UNGUIDED\unguided2b3.go"
Masukkan berat belanja di kedua kantong: 5 10
Sepeda motor pak Andi akan oleng: false
Masukkan berat belanja di kedua kantong: 55.6 70.2
Sepeda motor pak Andi akan oleng: true
Masukkan berat belanja di kedua kantong: 72.3 66.9
Sepeda motor pak Andi akan oleng: false
Masukkan berat belanja di kedua kantong: 59.5 98.7
Proses selesai.
PS C:\Users\lenovo\Documents\UNGUIDED>
```

## Deskripsi Program :

Program yang diberikan bertujuan untuk menghitung dan mengevaluasi berat belanjaan di dua kantong yang dibawa oleh sepeda motor. Pada awalnya, program akan meminta pengguna untuk memasukkan berat dari kedua kantong. Setelah menerima input, program akan memeriksa apakah berat yang dimasukkan valid; yaitu tidak boleh negatif dan total berat dari kedua kantong tidak boleh melebihi 150. Jika salah satu kondisi tersebut terpenuhi, program akan menghentikan proses dan mencetak pesan "Proses selesai." Jika berat yang dimasukkan valid, program akan menghitung selisih antara berat di kedua kantong. Jika selisih beratnya 9 atau lebih, maka sepeda motor dianggap akan oleng. Program ini ditampilkan kepada pengguna dengan pesan yang menunjukkan apakah sepeda motor akan oleng atau tidak. Dengan cara ini, program ini membantu pengguna untuk memastikan bahwa beban yang dibawa oleh sepeda motor berada dalam batas aman dan tidak akan menyebabkan ketidakstabilan saat berkendara.

## 2b\_4.) Source Code :

```
package main

import (
    "fmt"
    "math"
)

func main() {
    var K int

    // Minta pengguna untuk memasukkan nilai K
    fmt.Print("Nilai K= ")
```

```

fmt.Scanln(&K)

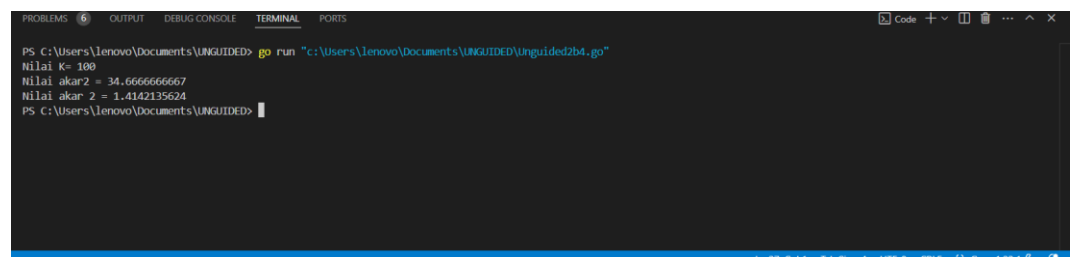
// Hitung akar2 dengan formula yang diberikan
akar2 := calculateAkar2(K)

// Tampilkan hasil
fmt.Printf("Nilai akar2 = %.10f\n", akar2)
fmt.Printf("Nilai akar 2 = %.10f\n", math.Sqrt(2))
}

// Fungsi untuk menghitung nilai akar2
func calculateAkar2(K int) float64 {
    return (1 + float64(K) + 3) / (1 + 2)
}

```

### Output :



```

PS C:\Users\lenovo\Documents\UNGUIDED> go run "c:\Users\lenovo\Documents\UNGUIDED\unguided2b4.go"
Nilai K= 100
Nilai akar2 = 34.6666666667
Nilai akar 2 = 1.4142135624
PS C:\Users\lenovo\Documents\UNGUIDED>

```

### Deskripsi Program :

Program yang diberikan bertujuan untuk menghitung dan menampilkan dua nilai: satu nilai berdasarkan input pengguna dan satu nilai tetap, yaitu akar kuadrat dari 2. Saat program dijalankan, pengguna diminta untuk memasukkan nilai integer yang disimpan dalam variabel K. Program kemudian menampilkan hasil perhitungan akar2 dengan format sepuluh angka desimal, menunjukkan keakuratan hasil yang diperoleh. Selain itu, program juga mencetak nilai akar kuadrat dari 2, yang tetap konstan, yaitu sekitar 1.4142135624, dalam format yang sama. Dengan cara ini, program memberikan pengguna dua informasi yang berguna: hasil perhitungan berdasarkan input dan nilai matematis tetap, sambil menunjukkan kemampuan bahasa Go dalam menangani input dan output dengan format yang tepat.

#### IV. UNGUIDED

##### 2C\_1.) Source Code :

```
package main

import (
    "fmt"
)

// Fungsi untuk menghitung biaya pengiriman berdasarkan
berat
func hitungBiayaPengiriman(beratGram int) (int, int, int) {
    beratKg := beratGram / 1000
    sisaGram := beratGram % 1000

    biayaPerKg := 10000
    biayaTambahan := calculateAdditionalCost(sisaGram,
beratKg)

    totalBiaya := (beratKg * biayaPerKg) + biayaTambahan
    return beratKg, sisaGram, totalBiaya
}

// Fungsi untuk menghitung biaya tambahan berdasarkan sisa
berat
func calculateAdditionalCost(sisaGram int, beratKg int) int {
    if beratKg > 10 {
        return 0
    } else if sisaGram >= 500 {
        return sisaGram * 5
    }
    return sisaGram * 15
}

func main() {
    var beratParcel int
    fmt.Print("Masukkan berat parcel (dalam gram): ")
}
```

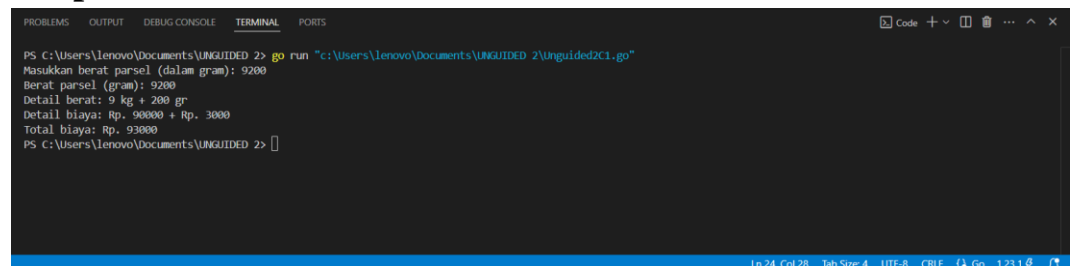
```

    fmt.Scan(&beratParcel)

    beratKg, sisaGram, biaya :=
hitungBiayaPengiriman(beratParcel)
    fmt.Printf("Berat parcel (gram): %d\n", beratParcel)
    fmt.Printf("Detail berat: %d kg + %d gr\n", beratKg,
sisaGram)
    fmt.Printf("Detail biaya: Rp. %d + Rp. %d\n",
beratKg*10000, biaya-(beratKg*10000))
    fmt.Printf("Total biaya: Rp. %d\n", biaya)
}

```

## Output :



```

PS C:\Users\lenovo\Documents\UNGUIDED 2> go run "c:\Users\lenovo\Documents\UNGUIDED 2\unguided2C1.go"
Masukkan berat parcel (dalam gram): 9200
Berat parcel (gram): 9200
Detail berat: 9 kg + 200 gr
Detail biaya: Rp. 90000 + Rp. 3000
Total biaya: Rp. 93000
PS C:\Users\lenovo\Documents\UNGUIDED 2>

```

## Deskripsi Program :

Program ini menghitung biaya pengiriman parcel berdasarkan berat yang dimasukkan dalam gram. Setelah pengguna memasukkan berat parcel, program mengonversi berat tersebut menjadi kilogram dan sisa gram. Biaya dasar dihitung dengan mengalikan berat dalam kilogram dengan tarif Rp. 10.000 per kilogram. Selain itu, biaya tambahan ditentukan berdasarkan sisa berat: jika sisa berat lebih dari atau sama dengan 500 gram, dikenakan biaya Rp. 5 per gram, sementara jika kurang dari 500 gram, biayanya Rp. 15 per gram. Program kemudian menampilkan detail berat dalam kilogram dan gram, biaya dasar, biaya tambahan, serta total biaya pengiriman. Dengan cara ini, pengguna dapat dengan mudah mengetahui total biaya yang harus dibayar untuk pengiriman parcel mereka.

## 2C\_2.) Source Code :

```
package main

import "fmt"

func main() {
    for {
        var nilaiAkhir float64

        // Meminta input nilai akhir dari pengguna
        fmt.Print("Nilai akhir mata kuliah (masukkan -1 untuk
keluar): ")
        fmt.Scanln(&nilaiAkhir)

        // Memeriksa apakah pengguna ingin keluar
        if nilaiAkhir == -1 {
            fmt.Println("Proses selesai.")
            break
        }

        // Menentukan grade berdasarkan nilai akhir
        nilaiHuruf := hitungNilaiHuruf(nilaiAkhir)

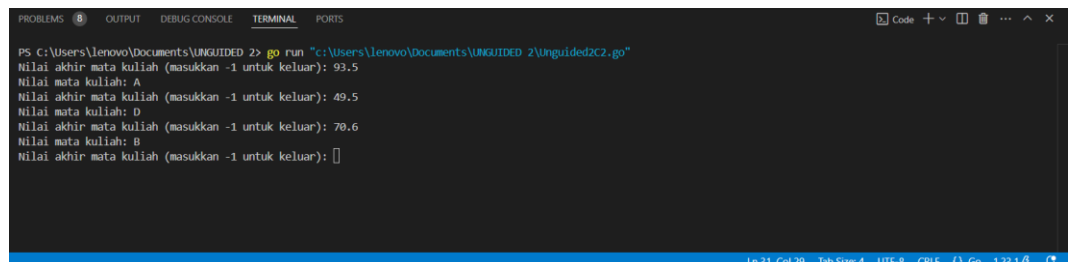
        // Menampilkan hasil grade
        fmt.Printf("Nilai mata kuliah: %v\n", nilaiHuruf)
    }
}

// Fungsi untuk menghitung nilai huruf berdasarkan nilai akhir
func hitungNilaiHuruf(nilai float64) string {
    if nilai > 80 {
        return "A"
    } else if nilai > 72.5 {
        return "AB"
    } else if nilai > 65 {
        return "B"
    } else if nilai > 57.5 {
        return "BC"
    } else if nilai > 50 {
        return "C"
    }
}
```



```
    } else if nilai > 40 {  
        return "D"  
    } else {  
        return "E"  
    }  
}
```

## Output :



```
PS C:\Users\lenovo\Documents\UNGUIDED 2> go run "c:\Users\lenovo\Documents\UNGUIDED 2\unguided2C2.go"  
Nilai akhir mata kuliah (masukkan -1 untuk keluar): 93.5  
Nilai mata kuliah: A  
Nilai akhir mata kuliah (masukkan -1 untuk keluar): 49.5  
Nilai mata kuliah: D  
Nilai akhir mata kuliah (masukkan -1 untuk keluar): 70.6  
Nilai mata kuliah: B  
Nilai akhir mata kuliah (masukkan -1 untuk keluar):
```

## Deskripsi Program :

Program ini adalah aplikasi sederhana dalam bahasa Go yang digunakan untuk menghitung dan menampilkan nilai huruf (grade) berdasarkan input nilai akhir dari pengguna. Pengguna diminta untuk memasukkan nilai akhir dalam bentuk angka desimal, dengan opsi untuk mengakhiri program dengan memasukkan -1. Jika nilai akhir yang dimasukkan valid, program akan memanggil fungsi `hitungNilaiHuruf` untuk menentukan grade sesuai dengan rentang nilai yang telah ditentukan: "A" untuk nilai di atas 80, "AB" untuk 72.5 hingga 80, "B" untuk 65 hingga 72.5, "BC" untuk 57.5 hingga 65, "C" untuk 50 hingga 57.5, "D" untuk 40 hingga 50, dan "E" untuk nilai di bawah 40. Setelah grade dihitung, program menampilkan hasilnya di konsol. Program ini berulang hingga pengguna memutuskan untuk keluar, sehingga memungkinkan pengguna untuk mengevaluasi beberapa nilai dalam satu sesi.

## 2C\_3.) Source Code :

```
package main
```

```

import (
    "fmt"
)

func main() {
    var bilangan int

    // Meminta input dari pengguna
    fmt.Print("Bilangan: ")
    fmt.Scanln(&bilangan)

    // Memeriksa apakah bilangan lebih besar dari 1
    if bilangan <= 1 {
        fmt.Println("Silakan masukkan bilangan bulat b >
1.")
        return
    }

    // Mencetak faktor dan memeriksa apakah bilangan
prima
    fmt.Print("Faktor: ")
    faktorPrima := true

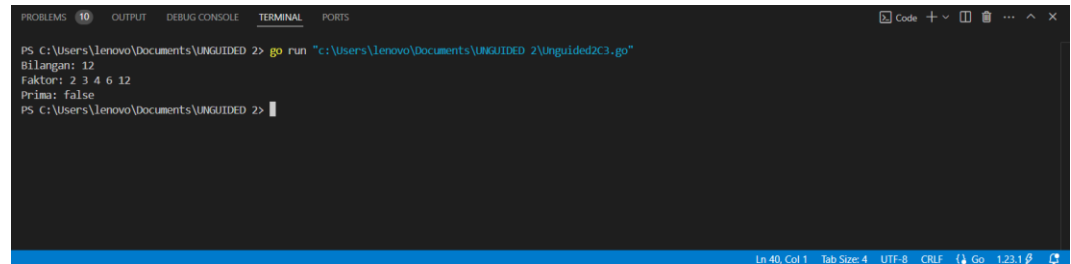
    for i := 2; i <= bilangan/2; i++ { // Mengurangi iterasi
hingga bilangan/2
        if bilangan%i == 0 {
            fmt.Print(i, " ")
            faktorPrima = false
        }
    }
    fmt.Print(bilangan) // Mencetak bilangan itu sendiri
sebagai faktor
    fmt.Println()

    // Menentukan apakah bilangan tersebut adalah bilangan
prima
    if faktorPrima {
        fmt.Println("Prima: true")
    } else {

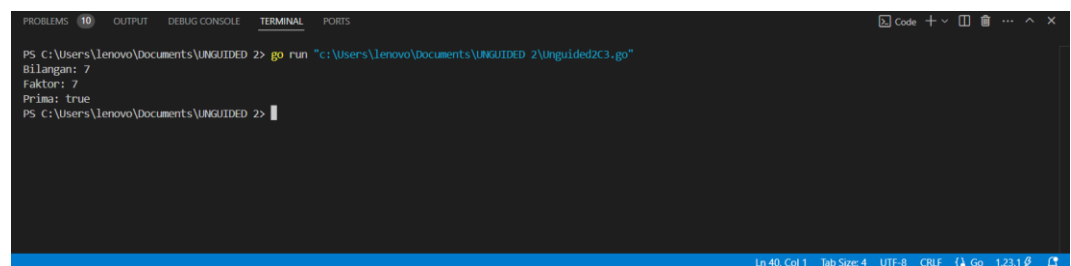
```

```
        fmt.Println("Prima: false")
    }
}
```

## Output :



```
PS C:\Users\lenovo\Documents\UNGUIDED 2> go run "c:\Users\lenovo\Documents\UNGUIDED 2\UnGUIDED2C3.go"
Bilangan: 12
Faktor: 2 3 4 6 12
Prima: false
PS C:\Users\lenovo\Documents\UNGUIDED 2>
```



```
PS C:\Users\lenovo\Documents\UNGUIDED 2> go run "c:\Users\lenovo\Documents\UNGUIDED 2\UnGUIDED2C3.go"
Bilangan: 7
Faktor: 7
Prima: true
PS C:\Users\lenovo\Documents\UNGUIDED 2>
```

## Deskripsi Program :

Program di atas meminta input dari pengguna berupa sebuah bilangan bulat, kemudian menghitung dan mencetak faktor-faktor dari bilangan tersebut. Di dalam kodingan diatas memasukan bilangan program menyatakan bahwa bilangan tersebut adalah bilangan prima. Output program mencakup faktor-faktor yang ditemukan dan status prima dari bilangan yang dimasukkan.