

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2  
MODUL II  
REVIEW STRUKTUR KONTROL**



**Oleh:**

**RINDI DELA NUR SAFITRI**

**2311102332**

**IF-11-07**

**S1 TEKNIK INFORMATIKA  
UNIVERSITAS TELKOM PURWOKERTO  
2024**

# I. DASAR TEORI

## 1.1 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.

Komentar, bukan bagian dari kode program, dan dapat ditulis di mana saja di dalam program:

- Satu baris teks yang diawali dengan garis miring ganda (`//`) s.d. akhir baris, atau.
- Beberapa baris teks yang dimulai dengan pasangan karakter `'/*'` dan diakhiri dengan `'*/'`.

```
1 // Setiap program utama dimulai dengan "package main"
2 package main
3
4 // Impor paket yang dibutuhkan, "fmt" berisi proses I/O standar
5 import "fmt"
6
7 // Kode program utama dalam "fungsi main"
8 func main() {
9     ...
10 }
```

## 2.2 Tipe Data dan Instruksi Dasar

### 1) Data dan Variabel

Variabel adalah nama dari suatu lokasi di memori, yang data dengan tipe tertentu dapat disimpan.

Nama variabel dimulai dengan huruf dan dapat diikuti dengan sejumlah huruf, angka, atau garisbawah.

Contoh: **ketemu**, **found**, **rerata**, **mhs1**, **data\_2**,...

Notasi tipe dasar	Tipe dalam Go	Keterangan
integer	int int8 int32 //rune int64 uint uint8 //byte uint32 uint64	bergantung platform 8 bit: -128..127 32 bit: -10 <sup>9</sup> ..10 <sup>9</sup> 64 bit: -10 <sup>19</sup> ..10 <sup>19</sup> bergantung platform 0..255 0..4294967295 0..(2 <sup>64</sup> -1)
real	float32 float64	32bit: -3.4E+38 .. 3.4E+38 64bit: -1.7E+308 .. 1.7E+308
boolean (atau logikal)	bool	false dan true
karakter	byte //uint8 rune //int32	tabel ASCII/UTF-8 tabel UTF-16
string	string	

- Tipe data yang umum tersedia adalah integer, real, boolean, karakter, dan string. Lihat tabel berikut ini untuk variasi tipe data yang disediakan dalam bahasa Go.
- Nilai data yang tersimpan dalam variabel dapat diperoleh dengan menyebutkan langsung nama variabelnya.
- Contoh: Menyebutkan nama found akan mengambil nilai tersimpan dalam memori untuk variabel found, pastinya.
- Informasi alamat atau lokasi dari variabel dapat diperoleh dengan menambahkan prefiks & di depan nama variabel tersebut.
- Contoh: &found akan mendapatkan alamat memori untuk menyimpan data pada found. Jika variabel berisi alamat memori, prefiks \* pada variabel tersebut akan memberikan nilai yang tersimpan dalam memori yang lokasinya disimpan dalam variabel tersebut.

### 2.3 Struktur Kontrol Perulangan

Go hanya mempunyai kata kunci for untuk semua jenis perulangan yang kita pelajari dalam notasi algoritma. Dua bentuk yang kita gunakan di sini adalah struktur while-loop dan repeat-until.

#### Bentuk perulangan dalam bahasa Go

```

1  for inisialisasi; kondisi; update {
2      // .. for-loop ala C
3      // .. ke-3 bagian opsional, tetapi ";" tetap harus ada
4  }
5  for kondisi {
6      // .. ulangi kode di sini selama kondisi terpenuhi
7      // .. sama seperti "for ; kondisi; {"
8  }
9  for {
10     // .. tanpa kondisi, berarti loop tanpa henti (perlu if-break)
11 }
12 for ndx, var := range slice/array {
13     // .. iterator mengunjungi seluruh isi slice/array
14     // .. pada setiap iterasi ndx diset indeks dan var diisi nilainya
15 }

```

## 2.4 Struktur Kontrol Percabangan

Untuk analisa kasus, bahasa Go mendukung dua bentuk percabangan, yaitu if-else dan switch-case.

### 1) Bentuk If-Else

Berikut ini bentuk-bentuk if-else yang mungkin dilakukan dalam bahasa Go. Semua bentuk di bawah merupakan satu instruksi if-else-endif saja (hanya satu endif). Bentuk if-else yang bersarang (dengan beberapa endif) dapat dibentuk dengan komposisi beberapa if-else-endif tersebut.

	Notasi algoritma	Penulisan dalam bahasa Go
1	if (kondisi) then	if kondisi {
2	.. kode untuk kondisi true	.. kode untuk kondisi true
3	endif	}
4	if (kondisi) then	if kondisi {
5	.. kode untuk kondisi true	.. kode untuk kondisi true
6	else	} else {
7	.. kode untuk kondisi false	.. kode untuk kondisi false
8	endif	}
9	if (kondisi-1) then	if kondisi_1 {
10	.. kode untuk kondisi-1 true	.. kode untuk kondisi_1 true
11	else if (kondisi-2) then	} else if kondisi_2 {
12	.. kode untuk kondisi-2 true	.. kode untuk kondisi_2 true
13	.. dst. dst.	.. dst. dst.
14	else	} else {
15	.. kode jika semua kondisi	.. kode jika semua kondisi
16	.. di atas false	.. di atas false
17	endif	}

### 2) Bentuk Switch-Case

Dalam bahasa Go ada dua variasi bentuk switch-case. Bentuk yang biasa digunakan adalah ekspresi ditulis pada perintah switch dan nilai ditulis dalam setiap label case-nya. Bentuk yang kedua mempunyai switch tanpa ekspresi, tetapi setiap case boleh berisi ekspresi boolean.

Tentunya bentuk yang kedua lebih bersifat umum, dan merupakan penyederhanaan bentuk (atau alias dari) susunan suatu if-elseif-..-else-endif.

	Notasi algoritma	Penulisan dalam bahasa Go
1	depend on ekspresi	switch ekspresi {
2	nilai_1:	case nilai_1:
3	.. kode jika ekspresi bernilai_1	.. kode jika ekspresi bernilai_1
4	nilai_2:	case nilai_2:
5	.. kode jika ekspresi bernilai_2	.. kode jika ekspresi bernilai_2
6	.. dst. dst.	.. dst. dst.
7	}	default:
8		.. kode jika tidak ada nilai
9		.. yang cocok dengan ekspresi
10		}
11	depend on (daftar variabel)	switch {
12	kondisi_1:	case kondisi_1:
13	.. kode jika ekspresi_1 true	.. kode jika ekspresi_1 true
14	kondisi_2:	case kondisi_2:
15	.. kode jika ekspresi_2 true	.. kode jika ekspresi_2 true
16	.. dst. dst.	.. dst. dst.
17	}	default:
18		.. jika tidak ada ekspresi
19		.. yang bernilai true
20		}

## II. GUIDED

### 2A

#### Guided 1

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?

```
1 package main
2 import "fmt"
3
4 func main() {
5     var (
6         satu, dua, tiga string
7         temp string
8     )
9     fmt.Print("Masukan input string: ")
10    fmt.Scanln(&satu)
11    fmt.Print("Masukan input string: ")
12    fmt.Scanln(&dua)
13    fmt.Print("Masukan input string: ")
14    fmt.Scanln(&tiga)
15    fmt.Println("Output awal = " + satu + " " + dua + " " + tiga)
16    temp = satu
17    satu = dua
18    dua = tiga
19    tiga = temp
20    fmt.Println("Output akhir = " + satu + " " + dua + " " + tiga)
21 }
```

## 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)
}
```

## Screenshot Program

A screenshot of a code editor showing a Go program and its execution output in a terminal. The code defines a main function that prompts the user for three strings, stores them in variables, swaps the first and second variables, and prints the results before and after the swap. The terminal output shows the user inputting 'pepaya', 'manggis', and 'anggur', followed by the program's output: 'Output awal = pepaya manggis anggur' and 'Output akhir = manggis anggur pepaya'.

```
1 package main
2 import "fmt"
3
4 func main() {
5     var (
6         satu, dua, tiga string
7         temp string
8     )
9     fmt.Print("Masukan input string: ")
10    fmt.Scanln(&satu)
11    fmt.Print("Masukan input string: ")
12    fmt.Scanln(&dua)
13    fmt.Print("Masukan input string: ")
14    fmt.Scanln(&tiga)
15    fmt.Println("Output awal = " + satu + " " + dua + " " + tiga)
16    temp = satu
17    satu = dua
```

PS C:\Users\ACER> go run "c:\Users\ACER\Downloads\go\guided1.go"

Masukan input string: pepaya  
Masukan input string: manggis  
Masukan input string: anggur  
Output awal = pepaya manggis anggur  
Output akhir = manggis anggur pepaya

## Penjelasan

Program ini dirancang untuk menerima tiga kata dari pengguna, kemudian menukar urutan kata-kata tersebut dan menampilkan hasil akhir. Program ini bekerja dengan cara mendeklarasikan variabel untuk menyimpan setiap kata, lalu melakukan pertukaran nilai antar variabel untuk mengubah

urutannya. Konsep dasar seperti variabel, input/output, dan penugasan nilai digunakan dalam program ini.

Program ini berfungsi untuk:

1. Meminta input: Program meminta pengguna untuk memasukkan tiga kata secara berurutan.
2. Menyimpan kata: Kata-kata yang dimasukkan disimpan dalam tiga variabel yang berbeda.
3. Menukar kata: Program melakukan pertukaran nilai antar variabel sehingga urutan kata-kata berubah.
4. Menampilkan hasil: Program menampilkan urutan kata-kata yang baru setelah ditukar.

## Guided 2

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). (Contoh input/output, Teks bergaris bawah adalah input dari user):

1	Tahun: <u>2016</u> Kabisat: true
2	Tahun: <u>2000</u> Kabisat: true
3	Tahun: <u>2018</u> Kabisat: false

## Source Code

```
package main

import "fmt"

func main() {
    var tahun int
    fmt.Println("Program menentukan false atau true nilai tahun kabisat")
    fmt.Print("Tahun:")
    fmt.Scanln(&tahun)

    if (tahun%4 == 0 && tahun%100 != 0) || (tahun%400 == 0) {
        fmt.Println("Kabisat: true")
    } else {
        fmt.Println("Kabisat: false")
    }
}
```

## Screenshot Program



```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var tahun int
7     fmt.Println("Program menentukan false atau true nilai tahun kabisat")
8     fmt.Print("Tahun: ")
9     fmt.Scanln(&tahun)
10
11     if (tahun%4 == 0 && tahun%100 != 0) || (tahun%400 == 0) {
12         fmt.Println("Kabisat: true")
13     } else {
14         fmt.Println("Kabisat: false")
15     }
16 }
17
```

PROBLEMS 30 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\ACER> go run "c:\Users\ACER\Downloads\go\guided2.go"

Program menentukan false atau true nilai tahun kabisat

Tahun: 2020

Kabisat: true

## Penjelasan

Program Go di atas adalah program untuk menentukan apakah suatu tahun merupakan tahun kabisat atau bukan. Berikut penjelasannya:

- Deklarasi variabel: Variabel tahun dideklarasikan dengan tipe int untuk menyimpan input dari pengguna.
- Input pengguna: Program menampilkan pesan untuk meminta input tahun dari pengguna menggunakan `fmt.Scanln(&tahun)`.
- Logika tahun kabisat: Program menggunakan kondisi untuk mengecek apakah tahun yang dimasukkan merupakan tahun kabisat: Sebuah tahun kabisat memenuhi salah satu dari dua kondisi berikut:



1. Tahun harus habis dibagi 4 dan tidak habis dibagi 100.
  2. Atau, tahun tersebut habis dibagi 400.
- Output: Jika salah satu kondisi di atas terpenuhi, maka program akan mencetak "Kabisat: true", menandakan bahwa tahun tersebut adalah tahun kabisat. Jika tidak, program mencetak "Kabisat: false", menandakan bukan tahun kabisat

### Guided 3

4. Dibaca nilai temperature dalam derajat Celcius. Nyatakan temperatur tersebut dalam Fahrenheit

$$Celsius = (Fahrenheit - 32) \times \frac{5}{9} \quad Reamur = Celsius \times \frac{4}{5} \quad Kelvin = (Fahrenheit + 459.67) \times \frac{5}{9}$$

(Contoh input/output, Teks bergaris bawah adalah input dari user):

Temperatur Celsius: <u>50</u> Derajat Fahrenheit: 122
--

Lanjutkan program di atas, sehingga temperatur dinyatakan juga dalam derajat Reamur dan Kelvin.

(Contoh input/output, Teks bergaris bawah adalah input dari user):

Temperatur Celsius: <u>50</u> Derajat Reamur: 40 Derajat Fahrenheit: 122 Derajat Kelvin: 323
---

## Source Code

```
package main

import (
    "fmt"
    "math"
)

func main() {
    var celsius float64

    fmt.Print("Temperatur Celsius: ")
    fmt.Scanln(&celsius)

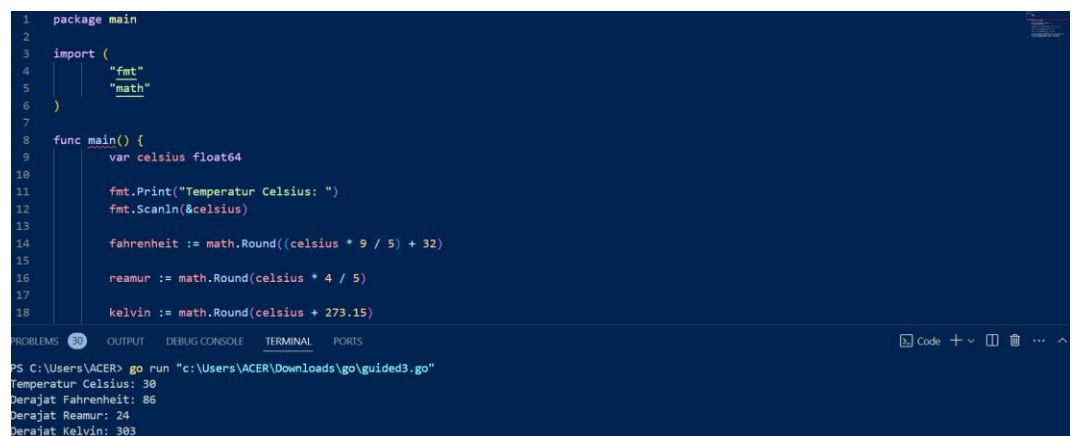
    fahrenheit := math.Round((celsius * 9 / 5) + 32)

    reamur := math.Round(celsius * 4 / 5)

    kelvin := math.Round(celsius + 273.15)

    fmt.Printf("Derajat Fahrenheit: %d\n", int(fahrenheit))
    fmt.Printf("Derajat Reamur: %d\n", int(reamur))
    fmt.Printf("Derajat Kelvin: %d\n", int(kelvin))
}
```

## Screenshot Program



```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func main() {
9     var celsius float64
10
11     fmt.Print("Temperatur Celsius: ")
12     fmt.Scanln(&celsius)
13
14     fahrenheit := math.Round((celsius * 9 / 5) + 32)
15
16     reamur := math.Round(celsius * 4 / 5)
17
18     kelvin := math.Round(celsius + 273.15)
19
20     fmt.Printf("Derajat Fahrenheit: %d\n", int(fahrenheit))
21     fmt.Printf("Derajat Reamur: %d\n", int(reamur))
22     fmt.Printf("Derajat Kelvin: %d\n", int(kelvin))
23 }
```

PS C:\Users\ACER> go run "c:\Users\ACER\Downloads\go\guided3.go"

Temperatur Celsius: 30  
Derajat Fahrenheit: 86  
Derajat Reamur: 24  
Derajat Kelvin: 303

## Penjelasan

Kode ini berfungsi sebagai konverter suhu. Program ini meminta pengguna untuk memasukkan suhu dalam Celcius, kemudian mengkonversi suhu tersebut ke Fahrenheit, Reamur, dan Kelvin. Hasil konversi kemudian ditampilkan ke layar.

#### Perhitungan Konversi:

- Fahrenheit: Menggunakan rumus konversi yang umum untuk mengubah Celsius menjadi Fahrenheit, kemudian dibulatkan menggunakan `math.Round` dan dikonversi ke tipe integer sebelum ditampilkan.
- Reamur: Menggunakan rumus konversi untuk mengubah Celsius menjadi Reamur, dibulatkan, dan dikonversi ke integer.
- Kelvin: Menggunakan rumus konversi untuk mengubah Celsius menjadi Kelvin, dibulatkan, dan dikonversi ke integer.

### III. UNGUIDED

#### 2B

##### Unguided 1

1. Siswa kelas IPA di salah satu sekolah menengah atas di Indonesia sedang mengadakan praktikum kimia. Di setiap percobaan akan menggunakan 4 tabung reaksi, yang mana susunan warna cairan di setiap tabung akan menentukan hasil percobaan. Siswa diminta untuk mencatat hasil percobaan tersebut. Percobaan dikatakan berhasil apabila susunan warna zat cair pada gelas 1 hingga gelas 4 secara berturut-turut adalah 'merah', 'kuning', 'hijau', dan 'ungu' selama 5 kali percobaan berulang.

Buatlah sebuah program yang menerima input berupa warna dari ke 4 gelas reaksi sebanyak 5 kali percobaan. Kemudian program akan menampilkan **true** apabila urutan warna sesuai dengan informasi yang warna sesuai dengan informasi yang diberikan pada paragraf sebelumnya, dan **false** untuk Informatics lab urutan warna lainnya.

Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

Percobaan 1:	<u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
Percobaan 2:	<u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
Percobaan 3:	<u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
Percobaan 4:	<u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
Percobaan 5:	<u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
BERHASIL: true				
Percobaan 1:	<u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
Percobaan 2:	<u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
Percobaan 3:	<u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
Percobaan 4:	<u>ungu</u>	<u>kuning</u>	<u>hijau</u>	<u>merah</u>
Percobaan 5:	<u>merah</u>	<u>kuning</u>	<u>hijau</u>	<u>ungu</u>
BERHASIL: false				

## Source Code

```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strings"
)

func main() {
    correctOrder := []string{"merah", "kuning", "hijau",
"ungu"}

    reader := bufio.NewReader(os.Stdin)
    success := true

    for i := 1; i <= 5; i++ {
        fmt.Printf("Percobaan %d: ", i)

        input, _ := reader.ReadString('\n')
        input = strings.TrimSpace(input)
        colors := strings.Split(input, " ")

        if len(colors) != len(correctOrder) {
            fmt.Println("Jumlah warna tidak sesuai")
            continue
        }

        for j := 0; j < len(colors); j++ {
            if colors[j] != correctOrder[j] {
                success = false
                break
            }
        }

        if !success {
            break
        }
    }

    if success {
        fmt.Println("BERHASIL: TRUE")
    } else {
        fmt.Println("BERHASIL: FALSE")
    }
}
```

## Screenshot Program

```
1 package main
2
3 import (
4     "bufio"
5     "fmt"
6     "os"
7     "strings"
8 )
9
10 func main() {
11     correctOrder := []string{"merah", "kuning", "hijau", "ungu"}
12
13     reader := bufio.NewReader(os.Stdin)
14     success := true
15
16     for i := 1; i <= 5; i++ {
17         fmt.Printf("Percobaan %d: ", i)
18         input, _ := reader.ReadString('\n')
19     }
20
21     BERHASIL := true
22 }
```

PS C:\Users\ACER> go run "c:\Users\ACER\Downloads\go\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

```
1 package main
2
3 import (
4     "bufio"
5     "fmt"
6     "os"
7     "strings"
8 )
9
10 func main() {
11     correctOrder := []string{"merah", "kuning", "hijau", "ungu"}
12
13     reader := bufio.NewReader(os.Stdin)
14     success := true
15
16     for i := 1; i <= 5; i++ {
17         fmt.Printf("Percobaan %d: ", i)
18         input, _ := reader.ReadString('\n')
19     }
20
21     BERHASIL := false
22 }
```

PS C:\Users\ACER> go run "c:\Users\ACER\Downloads\go\unguided1.go"

Percobaan 1: merah kuning hijau ungu  
Percobaan 2: merah kuning hijau ungu  
Percobaan 3: merah kuning hijau ungu  
Percobaan 4: hijau merah ungu kuning  
BERHASIL: FALSE

## Penjelasan

Program ini meminta pengguna untuk memasukkan urutan 4 warna ("merah", "kuning", "hijau", "ungu") dalam 5 percobaan. Setiap input dipisah dan dibandingkan dengan urutan yang benar. Jika pengguna salah memasukkan urutan warna pada salah satu percobaan, program akan berhenti dan menampilkan "BERHASIL: FALSE". Jika semua percobaan benar, akan ditampilkan "BERHASIL: TRUE".

### Validasi Input:

- Program memeriksa apakah jumlah warna yang dimasukkan pengguna sama dengan urutan warna yang benar (harus ada 4 warna).
- Kemudian, program memeriksa apakah setiap warna dalam input pengguna sesuai dengan urutan yang benar di `correctOrder`.

### Kontrol Keluaran:

- Jika pengguna salah dalam memasukkan warna pada salah satu percobaan, permainan akan dihentikan dan program menampilkan pesan "BERHASIL: FALSE".
- Jika pengguna memasukkan semua warna dengan benar dalam 5 percobaan, maka program akan mencetak "BERHASIL: TRUE".

## Unguided 2

2. Suatu pita (string) berisi kumpulan nama-nama bunga yang dipisahkan oleh spasi dan '-', contoh pita diilustrasikan seperti berikut ini.

**Pita: mawar - melati - tulip - teratai - kamboja - anggrek**

Buatlah sebuah program yang menerima input sebuah bilangan bulat positif (dan tidak nol) N, kemudian program akan meminta input berupa nama bunga secara berulang sebanyak N kali dan nama tersebut disimpan ke dalam pita.

(Petunjuk: gunakan operasi penggabungan string dengan operator "+").

Tampilkan isi pita setelah proses input selesai.

Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

<b>N: <u>3</u></b> Bunga 1: <b><u>Kertas</u></b> Bunga 2: <b><u>Mawar</u></b> Bunga 3: <b><u>Tulip</u></b> Pita: Kertas - Mawar - Tulip -	<b>N : <u>0</u></b> Pita :
---	-------------------------------

Modifikasi program sebelumnya, proses input akan berhenti apabila user mengetikkan 'SELESAI'. Kemudian tampilkan isi pita beserta banyaknya bunga yang ada di dalam pita

Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

Bunga 1: <b><u>Kertas</u></b>	Bunga 1: <b><u>SELESAI</u></b>
Bunga 2: <b><u>Mawar</u></b>	Pita :
Bunga 3: <b><u>Tulip</u></b>	Bunga: 0
Bunga 4: <b><u>SELESAI</u></b>	
Pita: Kertas - Mawar - Tulip -	
Bunga: 3	

### Source Code

```
package main

import "fmt"

func main() {
    var n int
    var bunga, pita string

    fmt.Print("Masukkan jumlah bunga: ")
    fmt.Scanln(&n)

    for i := 1; i <= n; i++ {
        fmt.Printf("Bunga %d: ", i)
        fmt.Scanln(&bunga)

        if bunga == "SELESAI" {
            break
        }


        pita += bunga + "-"
    }

    if len(pita) > 0 {
        pita = pita[:len(pita)-1]
    }

    fmt.Println("Pita:", pita)
}
```



## Screenshot Program



```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var n int
7     var bunga, pita string
8
9     fmt.Print("Masukkan jumlah bunga: ")
10    fmt.Scanln(&n)
11
12    for i := 1; i <= n; i++ {
13        fmt.Printf("Bunga %d: ", i)
14        fmt.Scanln(&bunga)
15
16        if bunga == "SELESAT" {
17            break
18        }
19    }
20
21    pita = bunga
22    for i := 1; i < n; i++ {
23        pita = pita + " " + bunga
24        fmt.Scanln(&bunga)
25    }
26
27    fmt.Println(pita)
28 }
```

PROBLEMS 30 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\VACER> go run "c:\Users\VACER\Downloads\go\unguided2.go"

Masukkan jumlah bunga: 4

Bunga 1: kertas

Bunga 2: mawar

Bunga 3: tulip

Bunga 4: SELESAT

Pita: kertas-mawar-tulip

## Penjelasan

Program di atas adalah program yang meminta input berupa sejumlah nama bunga dari pengguna. Pertama, program meminta pengguna untuk memasukkan jumlah bunga yang akan dimasukkan, lalu dalam loop `for`, pengguna diminta untuk memasukkan nama bunga satu per satu. Jika pengguna memasukkan kata "SELESAT", loop akan berhenti. Setiap nama bunga yang dimasukkan akan ditambahkan ke variabel `pita`, diikuti oleh tanda hubung (``-``).

Setelah loop selesai, program memeriksa apakah ada tanda hubung terakhir di variabel `pita`, dan jika ada, tanda hubung tersebut dihapus. Akhirnya, program mencetak string yang berisi rangkaian nama bunga yang dipisahkan oleh tanda hubung. Program ini memungkinkan pengguna untuk menginput sejumlah bunga dan menggabungkannya menjadi satu string dengan format tertentu.

## Unguided 3

3. Setiap hari Pak Andi membawa banyak barang belanjaan dari pasar dengan mengendarai sepeda motor. Barang belanjaan tersebut dibawa dalam kantong terpal di kiri-kanan motor. Sepeda motor tidak akan oleng jika selisih berat barang di kedua kantong sisi tidak lebih dari 9 kg.

Buatlah program Pak Andi yang menerima input dua buah bilangan real positif yang menyatakan berat total masing-masing isi kantong terpal. Program akan terus meminta input bilangan tersebut hingga salah satu kantong terpal berisi 9 kg atau lebih.

Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

```
Masukan berat belanjaan di kedua kantong: 5.5 1.0  
Masukan berat belanjaan di kedua kantong: 7.1 8.5  
Masukan berat belanjaan di kedua kantong: 2 6  
Masukan berat belanjaan di kedua kantong: 9 5.8  
Proses selesai.
```

Pada modifikasi program tersebut, program akan menampilkan true jika selisih kedua isi kantong lebih dari atau sama dengan 9 kg. Program berhenti memproses apabila total berat isi kedua kantong melebihi 150 kg atau salah satu kantong beratnya negatif.

Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

```
Masukan berat belanjaan di kedua kantong: 5 10  
Sepeda motor pak Andi akan oleng: false  
Masukan berat belanjaan di kedua kantong: 55.6 70.2  
Sepeda motor pak Andi akan oleng: true  
Masukan berat belanjaan di kedua kantong: 72.3 66.9  
Sepeda motor pak Andi akan oleng: false  
Masukan berat belanjaan di kedua kantong: 59.5 98.7  
Proses selesai.
```

## Source Code

```
package main

import "fmt"

func main() {
    var berat1, berat2 float64
    var selisih, total float64
    var oleng bool

    for {
        fmt.Print("Masukkan berat belanjaan di kedua
kantong: ")

        fmt.Scan(&berat1, &berat2)

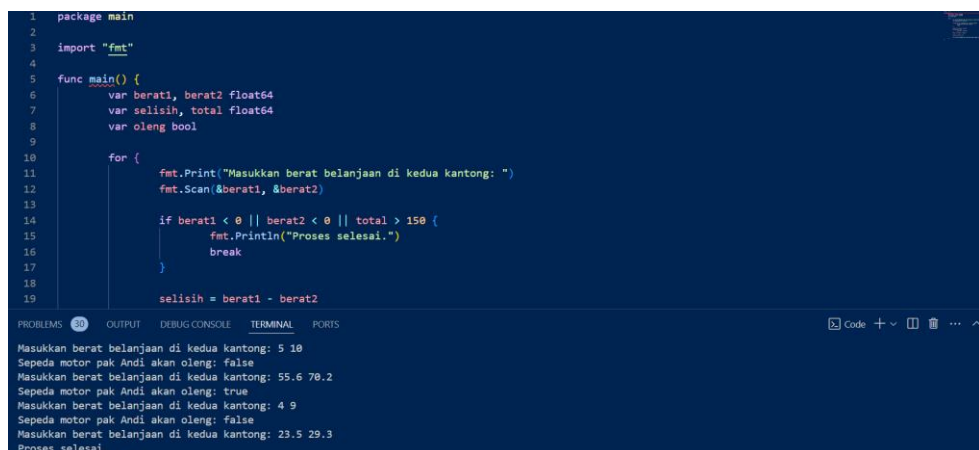
        if berat1 < 0 || berat2 < 0 || total > 150 {
            fmt.Println("Proses selesai.")
            break
        }

        selisih = berat1 - berat2
        if selisih < 0 {
            selisih = -selisih
        }
        total += berat1 + berat2

        oleng = selisih >= 9

        fmt.Printf("Sepeda motor pak Andi akan oleng:
%t\n", oleng)
    }
}
```

## Screenshot Program



```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var berat1, berat2 float64
7     var selisih, total float64
8     var oleng bool
9
10    for {
11        fmt.Print("Masukkan berat belanjaan di kedua kantong: ")
12        fmt.Scan(&berat1, &berat2)
13
14        if berat1 < 0 || berat2 < 0 || total > 150 {
15            fmt.Println("Proses selesai.")
16            break
17        }
18
19        selisih = berat1 - berat2
20
21        oleng = selisih >= 9
22
23        fmt.Printf("Sepeda motor pak Andi akan oleng: %t\n", oleng)
24    }
25}
```

PROBLEMS 30 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Masukkan berat belanjaan di kedua kantong: 5 10  
Sepeda motor pak Andi akan oleng: false  
Masukkan berat belanjaan di kedua kantong: 55.6 70.2  
Sepeda motor pak Andi akan oleng: true  
Masukkan berat belanjaan di kedua kantong: 4 9  
Sepeda motor pak Andi akan oleng: false  
Masukkan berat belanjaan di kedua kantong: 23.5 29.3  
Proses selesai.

## Penjelasan

Program di atas merupakan sebuah simulasi yang meminta pengguna untuk memasukkan berat belanjaan di dua kantong yang dibawa oleh Pak Andi saat menggunakan sepeda motor. Program ini akan terus meminta input berat dua kantong tersebut secara berulang hingga ada kondisi tertentu yang mengakhiri proses. Jika salah satu atau kedua berat belanjaan negatif, atau jika total berat belanjaan melebihi 150 kg, maka program akan berhenti dengan menampilkan pesan "Proses selesai." Program juga menghitung selisih berat antara kedua kantong dan menentukan apakah selisih tersebut cukup besar untuk menyebabkan sepeda motor oleng.

Untuk setiap iterasi, program menghitung selisih antara berat kantong pertama dan kedua. Jika selisihnya lebih dari atau sama dengan 9 kg, maka variabel `oleng` akan bernilai `true`, yang menandakan bahwa sepeda motor kemungkinan akan oleng akibat ketidakseimbangan beban. Selain itu, program menambahkan total berat dari kedua kantong ke variabel `total` untuk melacak berat kumulatif dari belanjaan yang dimasukkan. Proses ini akan terus berulang hingga salah satu kondisi batas terpenuhi (berat negatif atau total lebih dari 150 kg), setelah itu program berhenti.

## Unguided 4

4.

Diberikan sebuah persamaan sebagai berikut ini.

$$f(k) = \frac{(4k + 2)^2}{(4k + 1)(4k + 3)}$$

Buatlah sebuah program yang menerima input sebuah bilangan sebagai **K**, kemudian menghitung dan menampilkan nilai  $f(K)$  sesuai persamaan di atas.

Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

Nilai K = <u>100</u>
Nilai $f(K)$ = 1.0000061880

$\sqrt{2}$  merupakan bilangan irasional. Meskipun demikian, nilai tersebut dapat dihampiri dengan rumus berikut:

$$\sqrt{2} = \prod_{k=0}^{\infty} \frac{(4k+2)^2}{(4k+1)(4k+3)}$$

Modifikasi program sebelumnya yang menerima input integer  $K$  dan menghitung  $\sqrt{2}$  untuk  $K$  tersebut. Hampiran  $\sqrt{2}$  dituliskan dalam ketelitian 10 angka di belakang koma.

Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

1	Nilai K = <u>10</u> Nilai akar 2 = 1.4062058441
2	Nilai K = <u>100</u> Nilai akar 2 = 1.4133387072
3	Nilai K = <u>1000</u> Nilai akar 2 = 1.4141252651

### Source Code

```
package main

import (
    "fmt"
    "math"
)

func hitungAkarDua(k int) float64 {
    var hasil float64 = 1.0
    for i := 0; i <= k; i++ {
        pembilang := math.Pow(float64(4*i+2), 2)
        penyebut := float64(4*i+1) * float64(4*i+3)
        hasil *= pembilang / penyebut
    }
    return hasil
}

func main() {
    var k int

    fmt.Print("Nilai K= ")
    fmt.Scan(&k)

    akarDua := hitungAkarDua(k)
    fmt.Printf("Nilai akar 2= %.10f\n", akarDua)
}
```

## Screenshot Program



```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func hitungAkarDua(k int) float64 {
9     var hasil float64 = 1.0
10    for i := 0; i <= k; i++ {
11        pembilang := math.Pow(float64(4*i+2), 2)
12        penyebut := float64(4*i+1) * float64(4*i+3)
13        hasil *= pembilang / penyebut
14    }
15    return hasil
16 }
17
18 func main() {
19     var k int
20
21     // ... (user input and output) ...
22
23     go run "c:\Users\ACER\Downloads\go\unguided4.go"
24     Nilai K= 300
25     Nilai akar 2= 1.4139199441
26 }
```

## Penjelasan

Program ini menghitung nilai perkiraan akar kuadrat dari 2 menggunakan metode yang melibatkan deret matematika. Fungsi utama dari program ini adalah `hitungAkarDua(k int)`, yang menerima parameter `k` untuk menentukan berapa kali loop dijalankan dalam proses perhitungan. Fungsi ini menggunakan iterasi dan deret untuk menghitung nilai perkiraan akar kuadrat dari 2. Di dalam loop, setiap iterasi menghitung hasil dari pembilang dan penyebut, di mana pembilang adalah kuadrat dari  $4i+2$  dan penyebut adalah hasil kali  $4i+1$  dan  $4i+3$ . Nilai hasil akhir dikalikan dengan setiap hasil perbandingan pembilang dan penyebut pada setiap iterasi.

Di dalam fungsi `main`, program meminta pengguna memasukkan nilai `k` yang merupakan jumlah iterasi yang ingin dilakukan untuk menghitung perkiraan akar 2. Setelah nilai `k` diperoleh dari input, program memanggil fungsi `hitungAkarDua(k)` dan mencetak hasilnya dengan presisi 10 desimal. Semakin besar nilai `k`, semakin akurat nilai akar kuadrat dari 2 yang dihasilkan oleh program.

2C

## Unguided 5

1. PT POS membutuhkan aplikasi perhitungan biaya kirim berdasarkan berat parcel. Maka, **buatlah program BiayaPos untuk menghitung biaya pengiriman tersebut dengan ketentuan sebagai berikut!**

Dari berat parcel (dalam gram), harus dihitung total berat dalam kg dan sisanya (dalam gram). Biaya jasa pengiriman adalah Rp. 10.000,- per kg. Jika sisa berat tidak kurang dari 500 gram, maka tambahan biaya kirim hanya Rp. 5,- per gram

saja. Tetapi jika kurang dari 500 gram, maka tambahan biaya akan dibebankan sebesar Rp. 15,- per gram. Sisa berat (yang kurang dari 1kg) digratiskan biayanya apabila total berat ternyata lebih dari 10kg.

Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

1	Contoh #1 Berat parcel (gram): <u>8500</u> Detail berat: 8 kg + 500 gr Detail biaya: Rp. 80000 + Rp. 2500 Total biaya: Rp. 82500
2	Contoh #2 Berat parcel (gram): <u>9250</u> Detail berat: 9 kg + 250 gr Detail biaya: Rp. 90000 + Rp. 3750 Total biaya: Rp. 93750
3	Contoh #3 Berat parcel (gram): <u>11750</u> Detail berat: 11 kg + 750 gr Detail biaya: Rp. 110000 + Rp. 3750 Total biaya: Rp. 110000

## Source Code

```
package main

import "fmt"

func main() {
    var beratGram int

    fmt.Print("Berat parcel (gram): ")
    fmt.Scanln(&beratGram)

    beratKg := beratGram / 1000
    sisaGram := beratGram % 1000

    biayaDasar := beratKg * 10000

    var biayaTambahan int
    if sisaGram >= 500 {
        biayaTambahan = sisaGram * 5
    } else {
        biayaTambahan = sisaGram * 15
    }

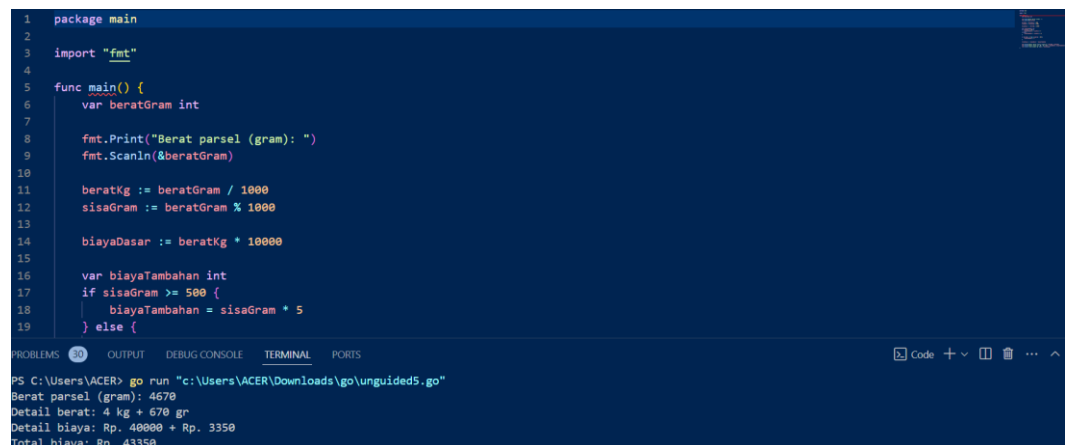
    if beratKg > 10 && sisaGram < 500 {
        biayaTambahan = 0
    }

    totalBiaya := biayaDasar + biayaTambahan

    fmt.Printf("Detail berat: %d kg + %d gr\n", beratKg,
sisaGram)
    fmt.Printf("Detail biaya: Rp. %d + Rp. %d\n", biayaDasar,
biayaTambahan)
    fmt.Printf("Total biaya: Rp. %d\n", totalBiaya)
}
```



## Screenshot Program



```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var beratGram int
7
8     fmt.Print("Berat parcel (gram): ")
9     fmt.Scanln(&beratGram)
10
11     beratKg := beratGram / 1000
12     sisaGram := beratGram % 1000
13
14     biayaDasar := beratKg * 10000
15
16     var biayaTambahan int
17     if sisaGram >= 500 {
18         biayaTambahan = sisaGram * 5
19     } else {
20         biayaTambahan = sisaGram * 15
21     }
22
23     totalBiaya := biayaDasar + biayaTambahan
24     fmt.Println("Detail berat: ", beratKg, " kg + ", sisaGram, " gr")
25     fmt.Println("Detail biaya: Rp. ", biayaDasar, " + Rp. ", biayaTambahan)
26     fmt.Println("Total biaya: Rp. ", totalBiaya)
27 }
```

PS C:\Users\ACER> go run "c:\Users\ACER\Downloads\go\unguided5.go"

Berat parcel (gram): 4670

Detail berat: 4 kg + 670 gr

Detail biaya: Rp. 40000 + Rp. 3350

Total biaya: Rp. 43350

## Penjelasan

Program di atas menghitung biaya pengiriman parcel berdasarkan berat yang dimasukkan dalam gram. Pertama, berat parcel diinputkan oleh pengguna dalam satuan gram dan kemudian dikonversi menjadi kilogram dan sisa gram menggunakan operasi pembagian dan modulus. Setelah itu, biaya dasar dihitung berdasarkan berat kilogram, di mana setiap kilogram dikenakan biaya Rp. 10.000. Selain itu, biaya tambahan dihitung berdasarkan berat sisa dalam gram: jika sisa gram lebih dari atau sama dengan 500, biaya tambahan dihitung Rp. 5 per gram, sedangkan jika kurang dari 500, biaya tambahan dihitung Rp. 15 per gram.

Program ini juga memiliki kondisi khusus: jika berat parcel lebih dari 10 kg dan sisa gram kurang dari 500, biaya tambahan akan dihapuskan (dijadikan nol). Setelah biaya dasar dan biaya tambahan dihitung, total biaya ditampilkan. Program memberikan rincian berat dalam kilogram dan gram, serta rincian biaya dasar dan biaya tambahan sebelum menampilkan total biaya pengiriman yang harus dibayar.

## Unguided 6

2. Diberikan sebuah nilai akhir mata kuliah (NAM) [0..100] dan standar penilaian nilai mata kuliah (NMK) sebagai berikut:

NAM	NMK
NAM > 80	A
72.5 < NAM <= 80	AB

$65 < \text{NAM} \leq 72.5$	B
$57.5 < \text{NAM} \leq 65$	BC
$50 < \text{NAM} \leq 57.5$	C
$40 < \text{NAM} \leq 50$	D
$\text{NAM} \leq 40$	E

Program berikut menerima input sebuah bilangan riil yang menyatakan NAM. Program menghitung NMK dan menampilkannya.

```

1 package main
2 import "fmt"
3 func main() {
4     var nam float64
5     var nmk string
6     fmt.Print("Nilai akhir mata kuliah: ")
7     fmt.Scanln(&nam)
8     if nam > 80 {
9         nam = "A"
10    }
11    if nam > 72.5 {
12        nam = "AB"
13    }
14    if nam > 65 {
15        nam = "B"
16    }
17    if nam > 57.5 {
18        nam = "BC"
19    }
20    if nam > 50 {
21        nam = "C"
22    }
23    if nam > 40 {
24        nam = "D"
25    } else if nam <= 40 {
26        nam = "E"
27    }
28    fmt.Println("Nilai mata kuliah: ", nmk)
29 }

```

Jawablah pertanyaan-pertanyaan berikut:

- Jika **nam** diberikan adalah 80.1, apa keluaran dari program tersebut? Apakah eksekusi program tersebut sesuai spesifikasi soal?
- Apa saja kesalahan dari program tersebut? Mengapa demikian? Jelaskan alur program seharusnya!
- 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.Scanln(&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.Println("Nilai mata kuliah:", nmk)
}
```

## Screenshot Program



```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var nam float64
7     var nmk string
8
9     fmt.Print("Nilai akhir mata kuliah: ")
10    fmt.Scanln(&nam)
11
12    if nam >= 80 {
13        nmk = "A"
14    } else if nam >= 72.5 {
15        nmk = "AB"
16    } else if nam >= 65 {
17        nmk = "B"
18    } else if nam >= 57.5 {
19        nmk = "BC"
20    }
21
22    fmt.Println("Nilai mata kuliah:", nmk)
23 }
```

PROBLEMS 30 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\ACER> go run "c:\Users\ACER\Downloads\go\unguided6.go"

Nilai akhir mata kuliah: 93.5

Nilai mata kuliah: A

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var nam float64
7     var nmk string
8
9     fmt.Print("Nilai akhir mata kuliah: ")
10    fmt.Scanln(&nam)
11
12    if nam >= 80 {
13        nmk = "A"
14    } else if nam >= 72.5 {
15        nmk = "AB"
16    } else if nam >= 65 {
17        nmk = "B"
18    } else if nam >= 57.5 {
19        nmk = "BC"
20    }
21
22    fmt.Println(nmk)
23 }
```

PROBLEMS 30 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\ACER> go run "c:\Users\ACER\Downloads\go\unguided6.go"  
Nilai akhir mata kuliah: 70.6  
Nilai mata kuliah: B

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var nam float64
7     var nmk string
8
9     fmt.Print("Nilai akhir mata kuliah: ")
10    fmt.Scanln(&nam)
11
12    if nam >= 80 {
13        nmk = "A"
14    } else if nam >= 72.5 {
15        nmk = "AB"
16    } else if nam >= 65 {
17        nmk = "B"
18    } else if nam >= 57.5 {
19        nmk = "BC"
20    }
21
22    fmt.Println(nmk)
23 }
```

PROBLEMS 30 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\ACER> go run "c:\Users\ACER\Downloads\go\unguided6.go"  
Nilai akhir mata kuliah: 49.5  
Nilai mata kuliah: D

## Penjelasan

Program ini bertujuan untuk mengkonversi nilai akhir mata kuliah (NAM) menjadi nilai huruf (NMK) sesuai dengan rentang nilai yang telah ditentukan. Namun, terdapat beberapa kesalahan logika dan sintaksis dalam program tersebut.

### Jawaban Pertanyaan

a. Jika `nam` diberikan adalah 80.1, apa keluaran dari program tersebut? Apakah eksekusi program tersebut sesuai spesifikasi soal?

Jika `nam` diberikan nilai 80.1, program akan mencetak A. Namun, eksekusi program ini tidak sepenuhnya sesuai dengan spesifikasi soal karena terdapat beberapa kesalahan logika yang akan dibahas pada poin b.

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

1. Kondisi `if` yang tumpang tindih: Beberapa kondisi `if` saling tumpang tindih, sehingga jika suatu nilai memenuhi syarat pada kondisi awal, kondisi selanjutnya tidak akan dievaluasi. Contohnya, jika `nam` adalah 80, maka kondisi pertama (`if nam > 80`) sudah terpenuhi dan program akan langsung mencetak A, tanpa mengecek kondisi lainnya.
2. Penggunaan operator perbandingan yang salah: Beberapa operator perbandingan digunakan secara tidak tepat. Misalnya, seharusnya menggunakan `>=` (lebih dari atau sama dengan) atau `<=` (kurang dari atau sama dengan) untuk mencakup batas-batas rentang nilai.
3. Sintaks yang salah: Ada beberapa kesalahan sintaksis seperti tanda kurung yang tidak lengkap dan penggunaan variabel yang belum dideklarasikan.

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

Perubahan yang dilakukan:

1. Kondisi `if else if`: Menggunakan struktur `if else if` untuk mengecek kondisi secara berurutan. Jika suatu kondisi terpenuhi, kondisi selanjutnya tidak akan dievaluasi.
2. Operator perbandingan: Menggunakan operator `>=` dan `<=` yang tepat untuk mencakup batas-batas rentang nilai.
3. Sintaks: Memperbaiki kesalahan sintaksis seperti tanda kurung dan penggunaan variabel.

## Unguided 7

3. Sebuah bilangan bulat **b** memiliki faktor bilangan **f** > 0 jika **f** habis membagi **b**. Contoh: 2 merupakan faktor dari bilangan 6 karena 6 habis dibagi 2.

Buatlah program yang menerima input sebuah bilangan bulat **b** dan **b** > 1. Program harus dapat mencari dan menampilkan semua faktor dari bilangan tersebut!

Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

Bilangan: <u>12</u> Faktor: 1 2 3 4 6 12	Bilangan: <u>7</u> Faktor: 1 7
---	-----------------------------------

Bilangan bulat **b** > 0 merupakan bilangan prima **p** jika dan hanya jika memiliki persis dua faktor bilangan saja, yaitu 1 dan dirinya sendiri.

Lanjutkan program sebelumnya. Setelah menerima masukan sebuah bilangan bulat **b** > 0. Program tersebut mencari dan menampilkan semua faktor bilangan tersebut. Kemudian, program menentukan apakah **b** merupakan bilangan prima.

Perhatikan contoh sesi interaksi program seperti di bawah ini (teks bergaris bawah adalah input/read):

Bilangan: <u>12</u> Faktor: 1 2 3 4 6 12 Prima: false	Bilangan: <u>7</u> Faktor: 1 7 Prima: true
---	--

## Source Code

```
package main

import "fmt"

func main() {
    var bilangan int

    fmt.Print("Bilangan: ")
    fmt.Scan(&bilangan)

    fmt.Print("Faktor: ")
    for i := 1; i <= bilangan; i++ {
        if bilangan%i == 0 {
            fmt.Printf("%d ", i)
        }
    }
    fmt.Println()

    var jumlahFaktor int = 0
    for i := 1; i <= bilangan; i++ {
        if bilangan%i == 0 {
            jumlahFaktor++
        }
    }

    if jumlahFaktor == 2 {
        fmt.Println("Prima: true")
    } else {
        fmt.Println("Prima: false")
    }
}
```

## Screenshot Program



```
1 package main
2
3 import "fmt"
4
5 func main() {
6     var bilangan int
7
8     fmt.Print("Bilangan: ")
9     fmt.Scan(&bilangan)
10
11     fmt.Print("Faktor: ")
12     for i := 1; i <= bilangan; i++ {
13         if bilangan%i == 0 {
14             fmt.Printf("%d ", i)
15         }
16     }
17     fmt.Println()
18
19     var jumlahFaktor int = 0
20
21     for i := 1; i <= bilangan; i++ {
22         if bilangan%i == 0 {
23             jumlahFaktor++
24         }
25     }
26
27     if jumlahFaktor == 2 {
28         fmt.Println("Prima: true")
29     } else {
30         fmt.Println("Prima: false")
31     }
32 }
```

PS C:\Users\ACER> go run "c:\Users\ACER\Downloads\go\unguided7.go"

Bilangan: 54

Faktor: 1 2 3 6 9 18 27 54

Prima: false

## Penjelasan

Program di atas adalah program yang meminta pengguna untuk memasukkan sebuah bilangan, kemudian mencetak faktor-faktor dari bilangan tersebut. Pertama, variabel `bilangan` dideklarasikan sebagai integer untuk menampung input dari pengguna. Program kemudian menggunakan `fmt.Scan` untuk membaca input tersebut. Setelah itu, program mencetak semua faktor dari bilangan tersebut dengan menggunakan perulangan `for`, di mana setiap bilangan dari 1 hingga bilangan yang diinputkan akan diuji apakah dapat membagi habis bilangan tersebut menggunakan operator modulus (%). Jika hasilnya 0, angka tersebut adalah faktor dan akan dicetak.

Selanjutnya, program menghitung jumlah faktor dari bilangan yang diinput dengan cara yang serupa, menggunakan perulangan `for` untuk mengecek berapa kali bilangan tersebut dapat dibagi habis. Hasilnya disimpan dalam variabel `jumlahFaktor`. Jika jumlah faktornya adalah 2 (artinya hanya 1 dan bilangan itu sendiri yang merupakan faktor), maka program mencetak bahwa bilangan tersebut adalah bilangan prima dengan pesan "Prima: true". Jika tidak, program mencetak "Prima: false", menunjukkan bahwa bilangan tersebut bukan bilangan prima.