LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK

MODUL 2

REVIEW STRUKTUR KONTROL

Oleh:

TRI PANJI UTOMO

2311102213

IF – 11- 02

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Dalam kerangka program yang ditulis dalam bahasa pemrograman Go,

program utama selalu mempunyai dua komponen berikut:

1. Package maln merupakan penanda bahwa file ini berisi program utama.
2. func main(berisi kode utama dari sebuah program Go.
3. Satu baris teks yang diawali dengan garis miring ganda ("l") s.d. akhir baris, atau.
4. Beberapa baris teks yang dimulai dengan pasangan karakter '" dan diakhiri dengan '"
5. Komentar, bukan bagian dari kode program, dan dapat ditulis di mana saja di dalam program.

```go
package main

import (
    "fmt"
    "math"
)

func main() {
    var jejari213, vol213, luas_kulit float64
    fmt.Print("jejari = ")
    fmt.Scan(&jejari213)
    vol213 = math.Pi * 4.0 / 3.0 * math.Pow(jejari213, 3)
    luas_kulit = math.Pi * 4 * math.Pow(jejari213, 2)
    fmt.Printf("Bola dengan jejari %v memiliki vol %.4f dan luas kulit %.4f", jejari213, vol213, luas_
}
```

Koding, Kompilast, dan Eksekusl Go

Koding

1. Tidak berbeda dengan penulisan program sumber dalam bahasa lain, program Go harus dibuat menggunakan penyunting teks dan disimpan dalam format teks, bukan dalam format dokumen (doc, docx, atau lainnya)
2. Setiap program go disimpan dalam file teks dengan ekstensi t go, dengan nama bebas Sebaiknya nama file adalah nama untuk program tersebut.
3. Setlap satu program lengkap Co 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.

Kompilast

Beberapa bahasa pemograman dirancang untuk diimplementasikan sebagai interpreter dan lainnya sebagai kompilator. Interpreter akan membaca setiap baris intruksi 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 eksekutabel, sehingga konsistensi penulisan (seperti penggunaan tipe data) sudah diperiksa sebelum eksekusi. Selain itu karna program dibuat menjadi eksekutabel lebih dahulu, proses optimasi dapat dilakukan sehingga program menjadi sangat efisien.

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

Variable

Variabel adalah nama dari suatu lokasi di memori, yang data dengan tipe tertentu dapat disimpan. ' Nama variabel dimulai dengan huruf dan dapat dikuti dengan sejumlah huruf, angka, atau garisbawah.

1. 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.
2. Nilal 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.
3. 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.
4. Jika variabel berisi alamat memori, prefiks * pada variabel tersebut akan memberikan nilai yang tersimpan dalam memori yang lokasinya disimpan dalam variabel tersebut.

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: -10^9..10^9 64 bit: -10^19..10^19 bergantung platform 0..255 0..4294967295 0..(2^64-1)
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	tabel tabel
string	string	

Struktur Kontrol

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

```
1
2
3
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,
13
14
15 }
```

Bentuk While-Loop Bentuk while-loop memastikan setiap kali memasuki loop, ada kondisi yang harus terpenuhi (benarltrue). Inijuga berarti saat keluar dari loop, maka nilai kondisi tersebut pasti salahlfalse!

	Notasi algoritma	Penulisan dalam bahasa Go
1	e <-	e :=
2	x <-	x :=
3	y <- 0.0	y := 0.0
4	y1 <- x	y1 := x
5	while y1-y > e or y1-y < -e do	for -y > e || -y < -e {
6	y <- y1	y = y1

Bentuk Repeat-Until Bentuk repeat-until di perulangan dilakukan terus menerus sampai kondisi keluar terpenuhi. Artinya selama kondisi belum terpenuhi (salahlfalse) maka perulangan akan terus dilakukan. Pada saat keluar dari loop maka nilai kondisi pasti benar/true!

	Notasi Algoritma	Penulisan dalam bahasa Go
1	repeat	for selesai:=false; !selesai; {
2	.. kode yang diulang	.. kode yang diulang
3	until (kondisi)	selesai = kondisi
4		}
5		for selesai:=false; !selesai;
6		selesai=kondisi {
7		..kode yang diulang
8		}
9		

Percabangan

Untuk analisa kasus, bahasa Go mendukung dua bentuk percabangan, yaitu if-else dan switch-case. 1) Bentuk If-Else Berikut ini bentuk-bentuk ifelse 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
8	endif	}
9		if kondisi_1 {
10		.. kode untuk kondisi_1 true
11		} else if kondisi_2 {
12		.. kode untuk kondisi
13		.. dst. dst.
14	else	} else {
15	.. kode jika semua kondisi	kode jika semua kondisi
16	di atas false	di atas false
17	endif	}

Bentuk Switch-Case Dalam bahasa Co ada dua variasi bentuk switchcase. 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.

	Notasi algoritma	Penulisan dalam bahasa Go

1
2
3
4
5
6
7
8
9
10
	depend on expresi
	nilai_1:
	.. kode jika ekspresi bernilai_1
	nilai_2:
	.. kode jika ekspresi bernilai_2
	.. dst. dst.
	}
		switch ekspresi {
		case nilai_1:
		.. kode jika ekspresi bernilai_1
		case nilai_2:
		.. kode jika ekspresi bernilai_2
		.. dst. dst.
		default:
		..kode jika tidak ada nilai
		..yang cocok dengan ekspresi
		}

11
12
13
14
15
16
17
18
19
20
	depend on (daftar variabel)
	kondisi_1:
	.. kode jika ekspresi_1 true
	kondisi_2:
	.. kode jika ekspresi_2 true
	.. dst. dst.
	}
		switch {
		case kondisi_1:
		.. kode jika ekspresi_1 true
		case kondisi_2:
		.. kode jika ekspresi _2 true
		.. dst. dst.
		default:
		..jika tidak ada ekspresi
		..yang bernilai true
		}

II. GUIDED

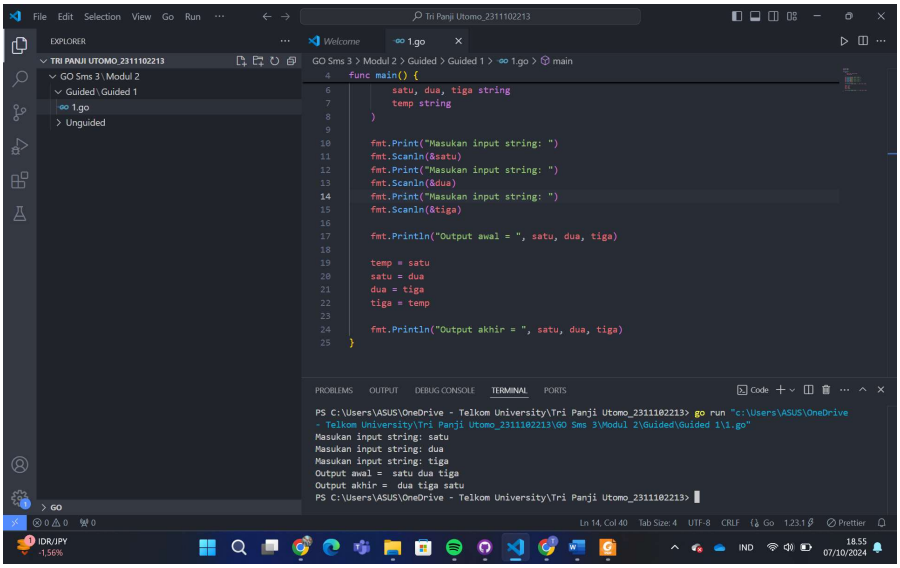
1. Guided 1

Source Code

```go
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



Deskripsi

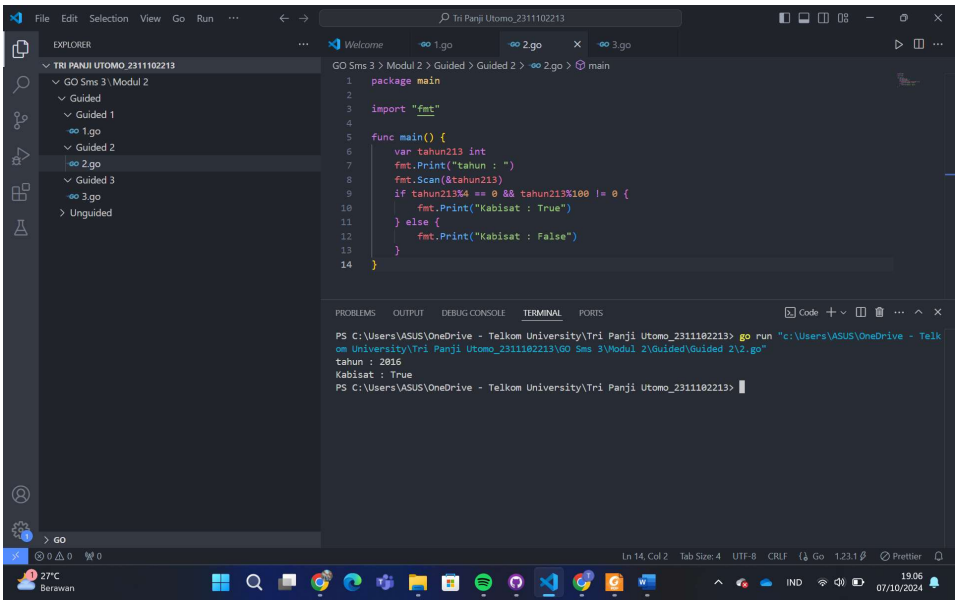
Program yang bertujuan untuk mengambil tiga input string dari pengguna, menampilkan urutan awal string tersebut, kemudian melakukan rotasi pada string, dan menampilkan urutan akhir setelah rotasi.

2. **Guided 2**

Source Code

```
package main

import "fmt"

func main() {
    var tahun213 int
    fmt.Print("tahun : ")
    fmt.Scan(&tahun213)
    if tahun213%4 == 0 && tahun213%100 != 0 {
        fmt.Print("Kabisat : True")
    } else {
        fmt.Print("Kabisat : False")
    }
}
```

Screenshot



Deskripsi

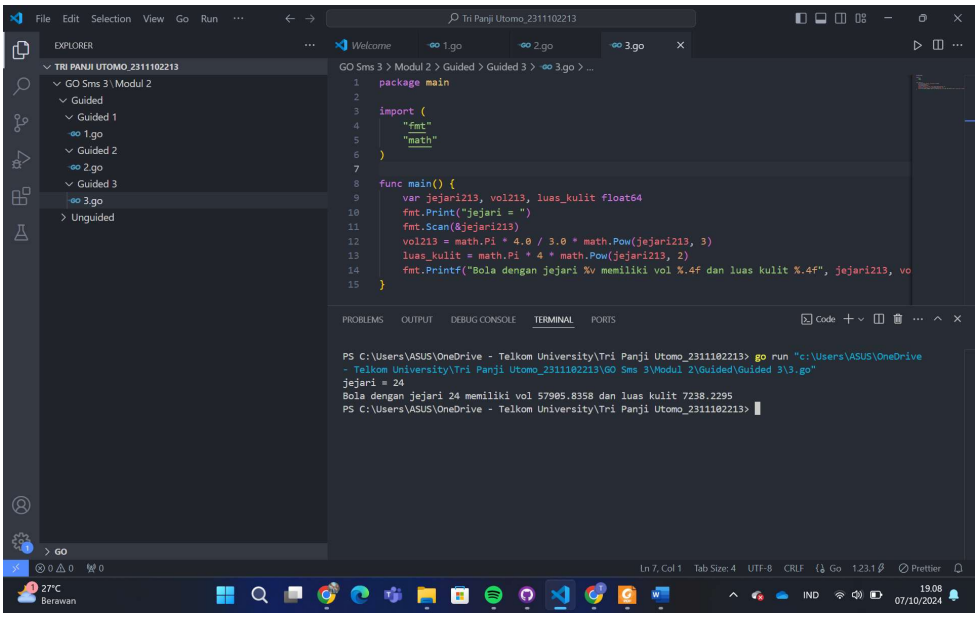
Proogram ini membantu menentukan apakah suatu tahun merupakan tahun kabisat yaitu habis dibagi 4 dan tidak habis dibagi 100. Jika kondisinya benar maka program akan mencetak true.

3. **Guided 3**

Source Code

```
package main

import (
    "fmt"
    "math"
)

func main() {
    var jejari213, vol213, luas_kulit float64
    fmt.Print("jejari = ")
    fmt.Scan(&jejari213)
    vol213 = math.Pi * 4.0 / 3.0 * math.Pow(jejari213, 3)
    luas_kulit = math.Pi * 4 * math.Pow(jejari213, 2)
    fmt.Printf("Bola dengan jejari %v memiliki vol %.4f dan luas kulit %.4f", jejari213, vol213, luas_kulit)
}
```

Screenshot



Deskripsi

Program ini bertujuan untuk menghitung volume dan luas permukaan sebuah bola berdasarkan input jejari (radius) yang diberikan oleh pengguna.

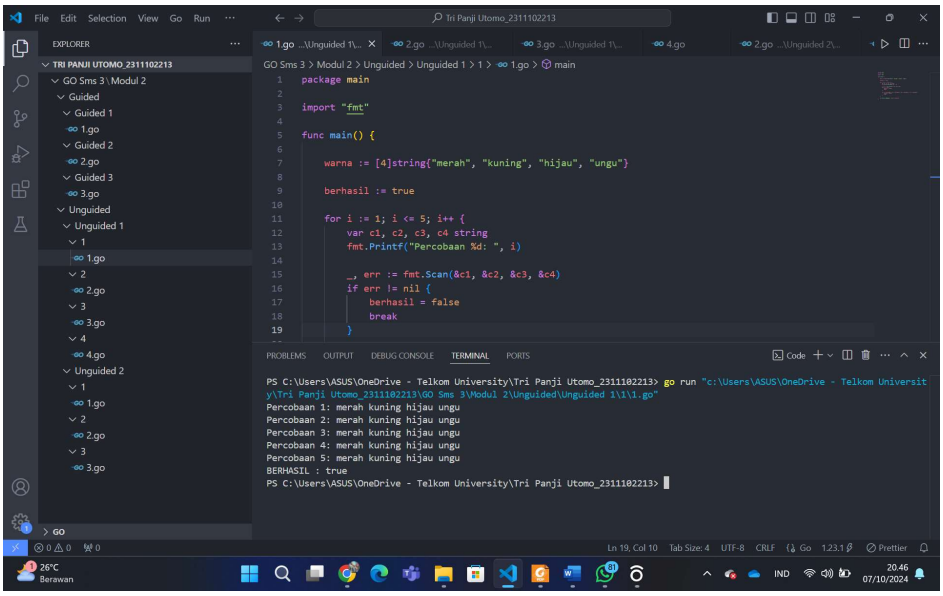
III. UNGUIDED

1. Unguided 1.1

Source Code

```
package main

import "fmt"

func main() {

    warna := [4]string{"merah", "kuning", "hijau", "ungu"}

    berhasil := true

    for i := 1; i <= 5; i++ {
        var c1, c2, c3, c4 string
        fmt.Printf("Percobaan %d: ", i)

        _, err := fmt.Scan(&c1, &c2, &c3, &c4)
        if err != nil {
            berhasil = false
            break
        }

        if c1 != warna[0] || c2 != warna[1] || c3 != warna[2] ||
c4 != warna[3] {
            berhasil = false
            break
        }
    }

    fmt.Printf("BERHASIL : %v\n", berhasil)
}
```

Screenshot



Deskripsi

Program ini bertujuan untuk memeriksa apakah pengguna dapat menginput empat warna dalam urutan yang benar selama lima percobaan.

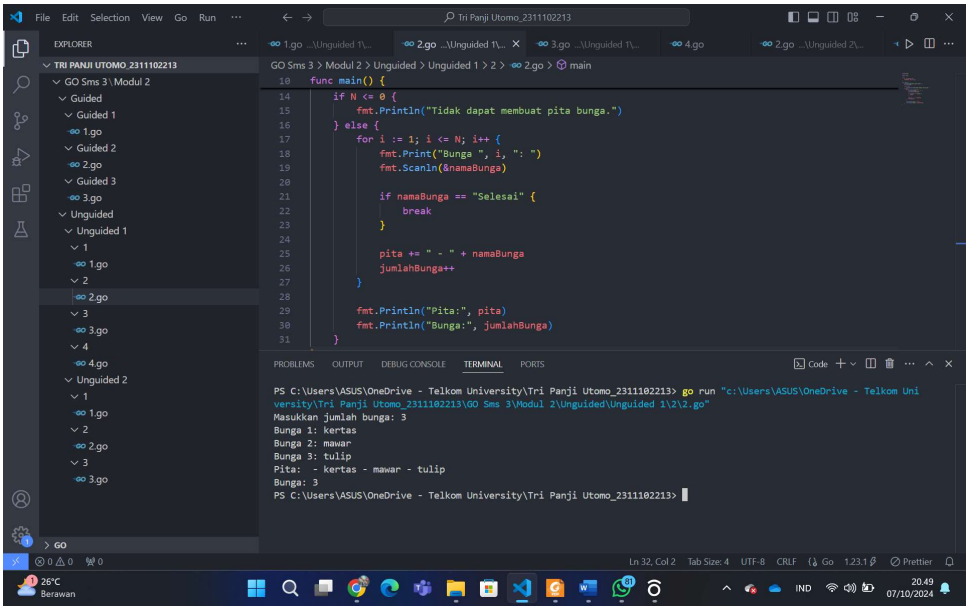
Unguided 1.2

Source Code

```go
package main

import "fmt"

var (
    N, jumlahBunga int
    namaBunga, pita string
)

func main() {
    fmt.Print("Masukkan jumlah bunga: ")
    fmt.Scanln(&N)

    if N <= 0 {
        fmt.Println("Tidak dapat membuat pita bunga.")
    } else {
        for i := 1; i <= N; i++ {
            fmt.Print("Bunga ", i, ": ")
            fmt.Scanln(&namaBunga)

            if namaBunga == "Selesai" {
                break
            }
```

```
            pita += " - " + namaBunga
            jumlahBunga++
        }

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

Screenshot



Deskripsi

Program ini dirancang untuk menerima input dari pengguna terkait jumlah bunga yang akan dimasukkan, kemudian menyusun nama-nama bunga tersebut menjadi sebuah string yang disebut "pita".

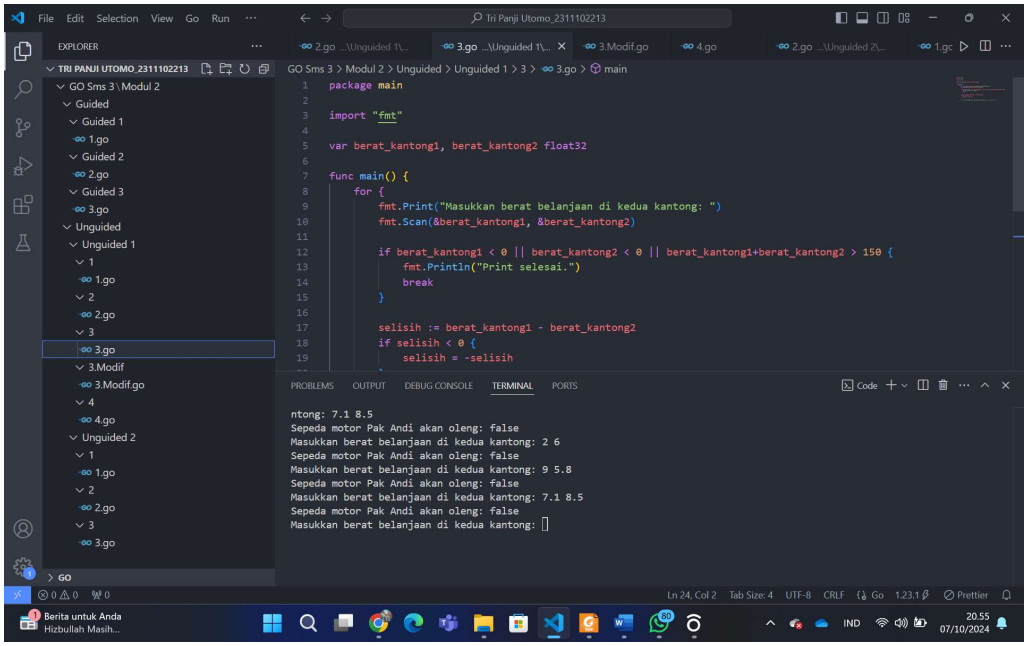
Unguided 1.3

Source Code

```
package main

import "fmt"

var berat_kantong1, berat_kantong2 float32

func main() {
    for {
        fmt.Print("Masukkan berat belanjaan di kedua kantong: ")
        fmt.Scan(&berat_kantong1, &berat_kantong2)
```

```
        if berat_kantong1 < 0 || berat_kantong2 < 0 ||
berat_kantong1+berat_kantong2 > 150 {
            fmt.Println("Print selesai.")
            break
        }

        selisih := berat_kantong1 - berat_kantong2
        if selisih < 0 {
            selisih = -selisih
        }

        fmt.Printf("Sepeda motor Pak Andi akan oleng: %t\n",
selisih >= 9)
    }
}
```

Screenshot



Deskripsi

Program ini bertujuan untuk mengevaluasi apakah beban belanjaan yang ditempatkan di dua kantong berbeda bisa menyebabkan sepeda motor Pak Andi oleng berdasarkan perbedaan berat antara kedua kantong.

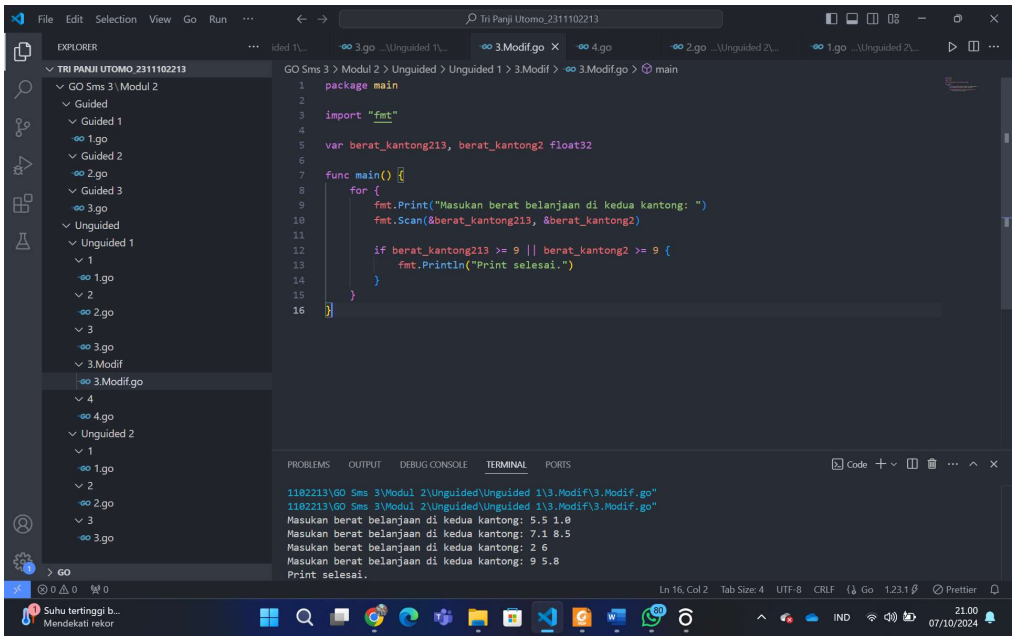
Unguided 1.3 Modifikasi

Source Code

```
package main

import "fmt"
```

```go
var berat_kantong213, berat_kantong2 float32

func main() {
    for {
        fmt.Print("Masukan berat belanjaan di kedua kantong: ")
        fmt.Scan(&berat_kantong213, &berat_kantong2)

        if berat_kantong213 >= 9 || berat_kantong2 >= 9 {
            fmt.Println("Print selesai.")
        }
    }
}
```

Screenshot



Deskripsi

Program ini bertujuan untuk mengevaluasi apakah beban belanjaan yang ditempatkan di dua kantong berbeda bisa menyebabkan sepeda motor Pak Andi oleng berdasarkan perbedaan berat antara kedua kantong.

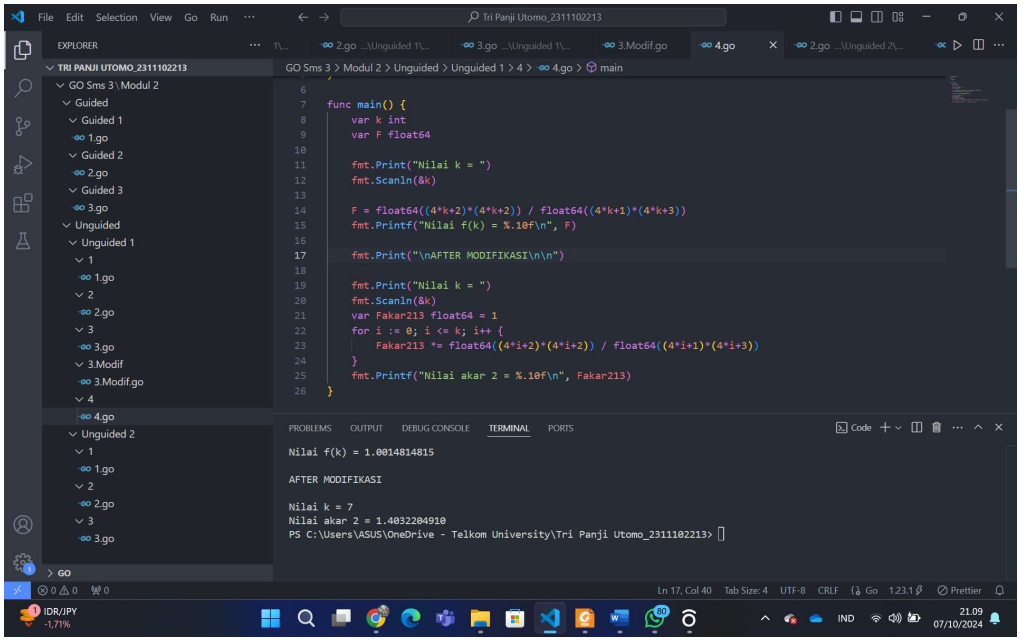
Unguided 1.4

Source Code

```go
package main

import (
    "fmt"
)

func main() {
```

```
    var k int
    var F float64

    fmt.Print("Nilai k = ")
    fmt.Scanln(&k)

    F = float64((4*k+2)*(4*k+2)) / float64((4*k+1)*(4*k+3))
    fmt.Printf("Nilai f(k) = %.10f\n", F)

    fmt.Print("\nAFTER MODIFIKASI\n\n")

    fmt.Print("Nilai k = ")
    fmt.Scanln(&k)
    var Fakar213 float64 = 1
    for i := 0; i <= k; i++ {
        Fakar213 *= float64((4*i+2)*(4*i+2)) /
float64((4*i+1)*(4*i+3))
    }
    fmt.Printf("Nilai akar 2 = %.10f\n", Fakar213)
}
```

Screenshot



Deskripsi

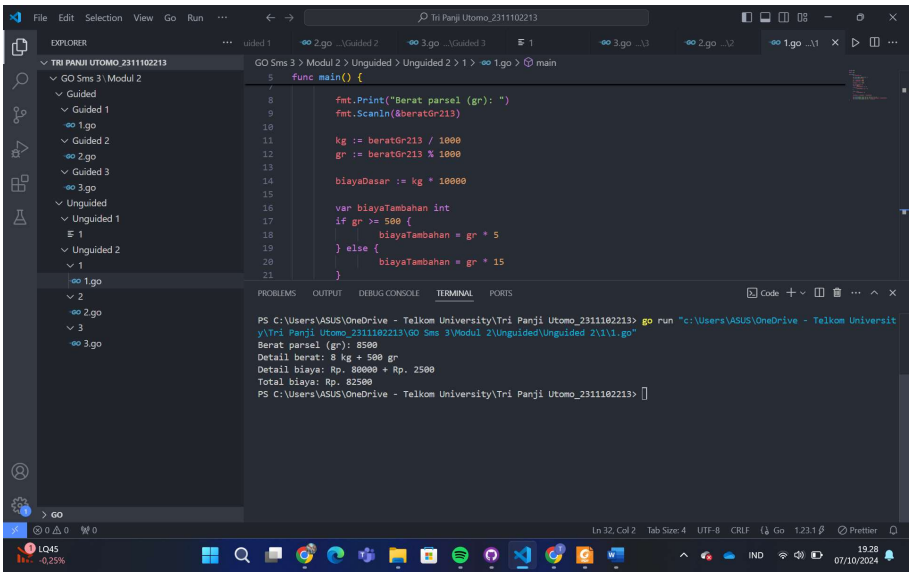
Program ini memiliki tujuan untuk menghitung dua nilai berbeda berdasarkan input K dari pengguna.

2. **Unguided 2.1**

Source Code

```
package main

import "fmt"

func main() {
        var beratGr213 int

        fmt.Print("Berat parsel (gr): ")
        fmt.Scanln(&beratGr213)

        kg := beratGr213 / 1000
        gr := beratGr213 % 1000

        biayaDasar := kg * 10000

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

        if kg > 10 {
                biayaTambahan = 0
        }

        totalBiaya := biayaDasar + biayaTambahan

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

Screenshot



Deskripsi

Program ini digunakan untuk menghitung biaya pengiriman parsel berdasarkan beratnya dalam gram.

Unguided 2.2

Source Code

```go
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)
}
```

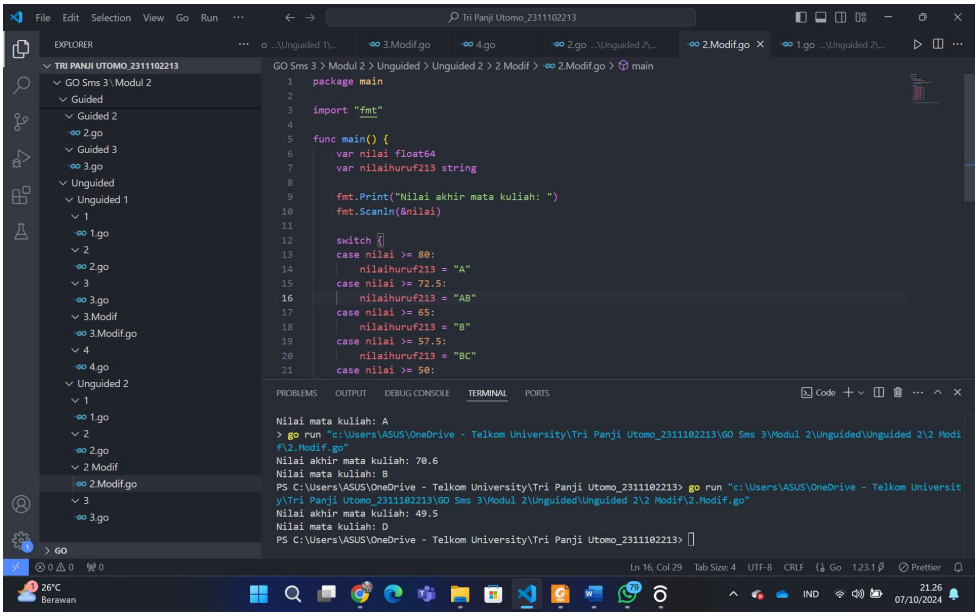
a. Jika nilai yang diberikan adalah 80.1, apa keluaran dari program tersebut? Apakah eksekusi program tersebut sesuai spesifikasi soal? Jika nilai yang diinputkan adalah 80.1, maka output "Nilai mata kuliah: A"yang sesuai sesuai dengan spesifikasi soal karena nilai 80.1 berada dalam rentang nilai untuk mendapatkan nilai huruf A (≥ 80).
b. Apa saja kesalahan dari program tersebut? Mengapa demikian? Jelaskan alur program seharusnya! Jadia lur yang seharusnya yaitu meminta input nilai akhir mata kuliah, membandingkan nilai tersebut dengan rentang nilai yang telah ditentukan, menentukan nilai huruf yang sesuai, mencetak nilai huruf ke layar.
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 nilai float64
    var nilaihuruf213 string

    fmt.Print("Nilai akhir mata kuliah: ")
    fmt.Scanln(&nilai)

    switch {
    case nilai >= 80:
        nilaihuruf213 = "A"
    case nilai >= 72.5:
        nilaihuruf213 = "AB"
    case nilai >= 65:
        nilaihuruf213 = "B"
    case nilai >= 57.5:
        nilaihuruf213 = "BC"
    case nilai >= 50:
        nilaihuruf213 = "C"
    case nilai >= 40:
        nilaihuruf213 = "D"
    default:
        nilaihuruf213 = "E"
    }
```

```
    fmt.Printf("Nilai mata kuliah: %s\n", nilaihuruf213)
}
```

Screenshot

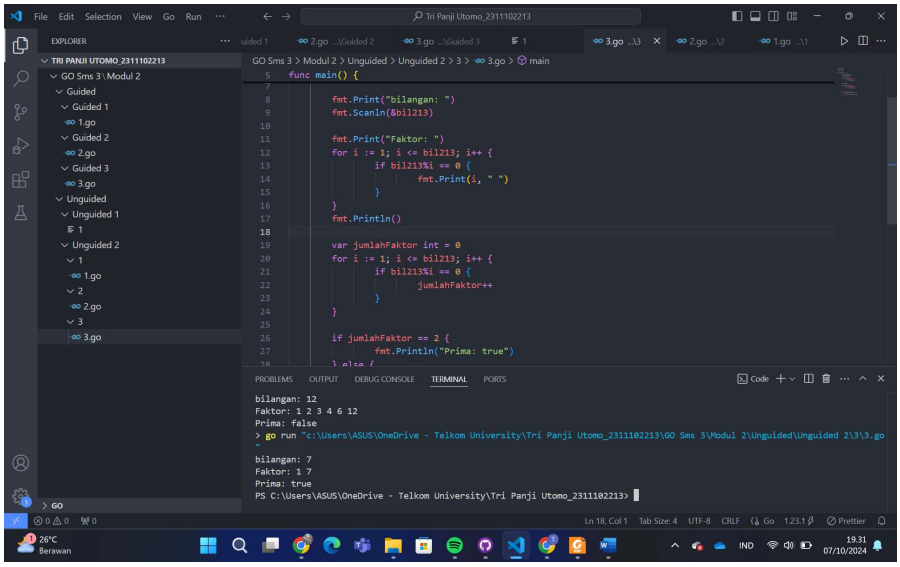


Unguided 2.3
Source Code

```
package main

import "fmt"

func main() {
        var bil213 int

        fmt.Print("bilangan: ")
        fmt.Scanln(&bil213)

        fmt.Print("Faktor: ")
        for i := 1; i <= bil213; i++ {
                if bil213%i == 0 {
                        fmt.Print(i, " ")
                }
        }
        fmt.Println()

        var jumlahFaktor int = 0
        for i := 1; i <= bil213; i++ {
                if bil213%i == 0 {
                        jumlahFaktor++
```

```
                }
        }

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

Screenshot



Deskripsi

Program ini bertujuan untuk menentukan faktor-faktor dari sebuah bilangan bulat yang diinput oleh user, sekaligus memeriksa apakah bilangan tersebut merupakan bilangan prima.