LAPORAN PRAKTIKUM ALGORITMA

DAN PEMROGRAMAN 1

MODUL 12

WHILE-LOOP

Disusun oleh:

RIZKY TABRIZ DEANOVA

109082500177

S1IF-13-07

Asisten Praktikum

Adithana Dharma Putra

Apri Pandu Wicaksono

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

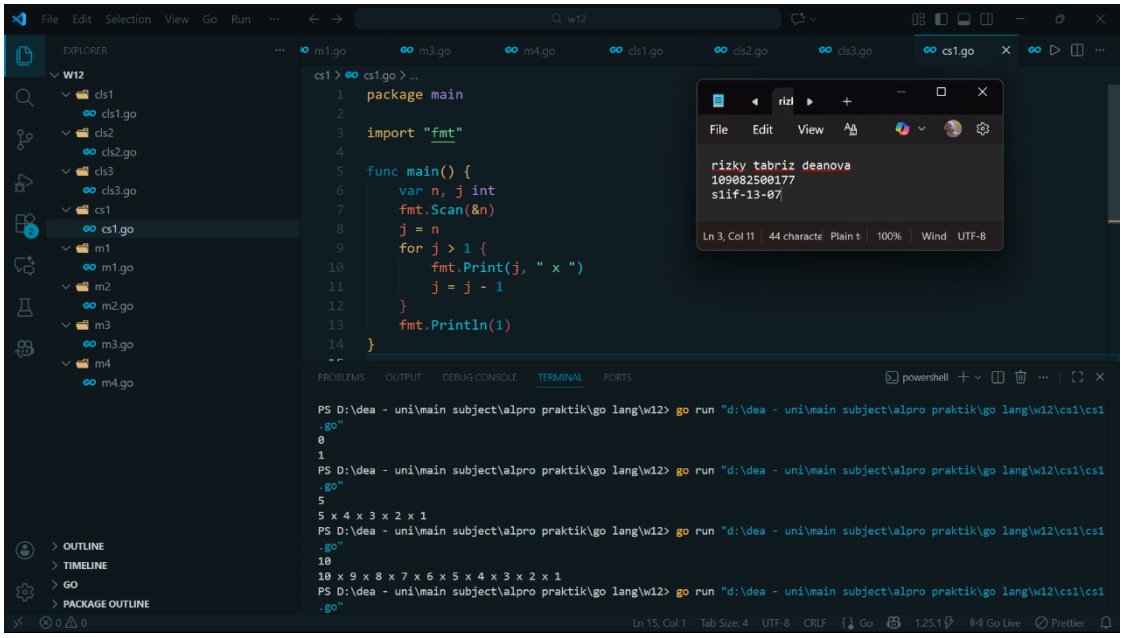
LATIHAN KELAS – GUIDED

1. Guided 1

Source Code

```
package main
import "fmt"
func main() {
    var n, j int
    fmt.Scan(&n)
    j = n
    for j > 1 {
        fmt.Print(j, " x ")
        j = j - 1
    }
    fmt.Println(1)
}
```

Screenshoot program



Deskripsi program

`package main` sebagai penanda bahwa program ini merupakan program *executable* atau program yang dapat dieksekusi (dijalankan).

`import ("fmt")` yang digunakan untuk memasukkan library `"fmt"` untuk dipakai menjalankan program nantinya.

`func main () {}` yang berperan sebagai penanda tempat program mulai berjalan, seluruh kode ataupun instruksi yang terdapat di dalam kurung `{}` setelah `func main ()` nantinya akan dieksekusi atau dijalankan secara sistematis (berurutan).
`var` dalam program adalah sebagai wadah utama untuk penyimpanan data sebelum nanti dibagi lagi sesuai tipe data.
`fmt.Scan` yang dalam bahasa pemrograman Go digunakan untuk menjeda hasil output sejenak guna membaca input user, yang kemudian disimpan di dalam variabel.
`fmt.Print` yang dalam bahasa pemrograman Go digunakan sebagai instruksi untuk mencetak atau menampilkan output.
`fmt.Println` yang dalam bahasa pemrograman Go digunakan sebagai instruksi untuk mencetak atau menampilkan output di garis baru.
`for` berfungsi sebagai perulangan, yang merupakan proses mengulang-ulang eksekusi blok kode di dalam perintah `for` tanpa henti selama kondisi yang diperintahkan terpenuhi, dalam blok kode `for` juga biasanya terdapat variabel penanda kapan perulangan akan diberhentikan.

Yang dilakukan program di dalam gambar di atas adalah **Menampilkan Deret Faktorial Dari Suatu Bilangan** yang mana terlihat melalui kode dari **baris 9 hingga baris 11**.

`for j > 1 {` → Memeriksa apakah kondisi *value* dalam variabel j lebih dari 1, jika lebih dari 1 atau ya, maka operasi dalam perulangan for akan dijalankan

`fmt.Print(j, "x")` → Mencetak nilai j yang sekarang dilanjutkan dengan alfabet x. Permisalahan output: 10 x

`j = j - 1` → Pembaruan *value* atau nilai variabel j setelah dilakukan operasi perulangan yang akan berulang hingg kondisi tidak sesuai dengan kondisi dalam perulangan

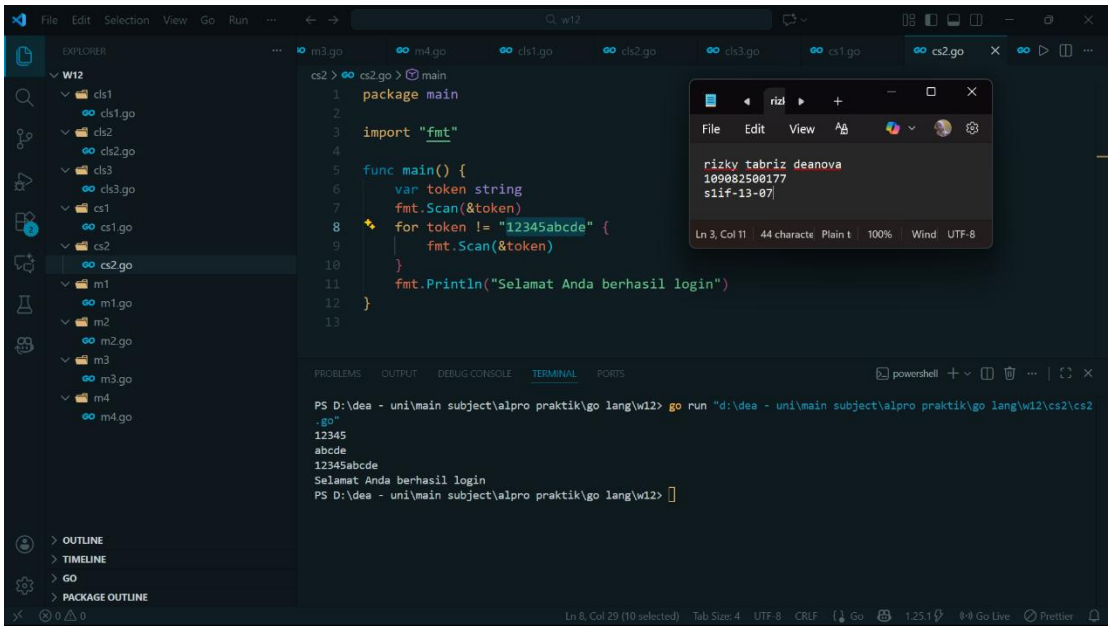
Runtutan Eksekusi:
1. Menunggu user melakukan input ke dalam variabel n yang akan dibaca lalu dimasukkan ke dalam variabel n
2. Melakukan deklarasi value untuk nilai j sama dengan nilai n
3. Memeriksa kondisi nilai variabel j sesuai dengan kondisi perulangan atau tidak, jika ya, maka akan dilanjutkan ke dalam operasi perulangan
4. Mencetak nilai j yang diikuti variabel x dan setelahnya nilai j yang sekarang akan dikurangi dengan nilai 1, lalu hasil operasi pengurangan tersebut akan dimasukkan ke dalam nilai variabel j yang terbaru. Hal ini seperti update dan akan terus berulang hingga kondisi nilai j tidak sesuai dengan kondisi dalam perulangan
5. Jika kondisi tidak sesuai dengan kondisi dalam perulangan sedari awal, maka operasi perulangan akan dihiarukan dan langsung beralih ke pencetakan nilai 1

2. Guided 2

Source Code

```
package main
import "fmt"
func main() {
    var token string
    fmt.Scan(&token)
    for token != "12345abcde" {
        fmt.Scan(&token)
    }
    fmt.Println("Selamat Anda berhasil login")
}
```

Screenshoot program



Deskripsi program

`package main` sebagai penanda bahwa program ini merupakan program *executable* atau program yang dapat dieksekusi (dijalankan).

`import ("fmt")` yang digunakan untuk memasukkan library `"fmt"` untuk dipakai menjalankan program nantinya.

`func main () {}` yang berperan sebagai penanda tempat program mulai berjalan, seluruh kode ataupun instruksi yang terdapat di dalam kurung `{}` setelah `func main ()` nantinya akan dieksekusi atau dijalankan secara sistematis (berurutan).

`var` dalam program adalah sebagai wadah utama untuk penyimpanan data sebelum nanti dibagi lagi sesuai tipe data.

`fmt.Scan` yang dalam bahasa pemrograman Go digunakan untuk menjeda hasil output sejenak guna membaca input user, yang kemudian disimpan di dalam variabel.
`fmt.Println` yang dalam bahasa pemrograman Go digunakan sebagai instruksi untuk mencetak atau menampilkan output di garis baru.
`for` berfungsi sebagai perulangan, yang merupakan proses mengulang-ulang eksekusi blok kode di dalam perintah `for` tanpa henti selama kondisi yang diperintahkan terpenuhi, dalam blok kode `for` juga biasanya terdapat variabel penanda kapan perulangan akan diberhentikan.

Yang dilakukan program di dalam gambar di atas adalah **Memeriksa Kesesuaian Input Token** yang mana terlihat melalui kode dari **baris 8 hingga baris 11**.

`for token != "123345abcde" {` → Memeriksa apakah kondisi *value* dalam variabel token tidak sama dengan abcde12345, jika tidak sama dengan atau ya, maka operasi dalam perulangan for akan dijalankan

`fmt.Scan(&token)` → Menunggu input baru dari user dan memasukkan input terbaru sebagai nilai terbaru dalam variabel token

`fmt.Println("Selamat Anda berhasil login")` → Mencetak output jika input token sudah sesuai dengan ketentuan

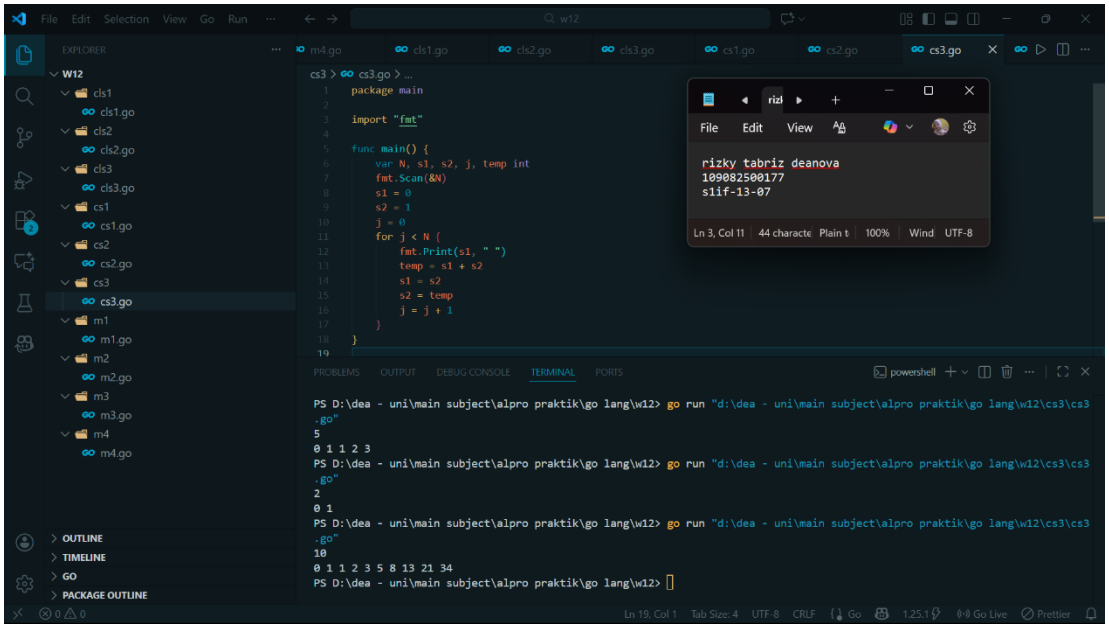
Runtutan Eksekusi:
1. Menunggu user melakukan input ke dalam variabel token yang akan dibaca lalu dimasukkan ke dalam variabel token
2. Memeriksa kondisi nilai variabel token sesuai dengan kondisi perulangan atau tidak, jika ya, maka akan dilanjutkan ke dalam operasi perulangan
3. Mencetak output ”Selamat Anda berhasil login” jika input token sudah sesuai dengan ketentuan
4. Jika kondisi tidak sesuai dengan kondisi dalam perulangan sedari awal, maka operasi perulangan akan dihiarukan dan langsung beralih ke pencetakan output ”Selamat Anda berhasil login”

3. Guided 3

Source Code

```
package main
import ("fmt")
func main() {
    var N, s1, s2, j, temp int
    fmt.Scan(&N)
    s1 = 0
    s2 = 1
    j = 0
    for j < N {
        fmt.Print(s1, " ")
        temp = s1 + s2
        s1 = s2
        s2 = temp
        j = j + 1
    }
}
```

Screenshoot program



Deskripsi program

`package main` sebagai penanda bahwa program ini merupakan program *executable* atau program yang dapat dieksekusi (dijalankan).

`import ("fmt")` yang digunakan untuk memasukkan library `"fmt"` untuk dipakai menjalankan program nantinya.
`func main () {}` yang berperan sebagai penanda tempat program mulai berjalan, seluruh kode ataupun instruksi yang terdapat di dalam kurung `{}` setelah `func main ()` nantinya akan dieksekusi atau dijalankan secara sistematis (berurutan).
`var` dalam program adalah sebagai wadah utama untuk penyimpanan data sebelum nanti dibagi lagi sesuai tipe data.
`fmt.Scan` yang dalam bahasa pemrograman Go digunakan untuk menjeda hasil output sejenak guna membaca input user, yang kemudian disimpan di dalam variabel.
`fmt.Print` yang dalam bahasa pemrograman Go digunakan sebagai instruksi untuk mencetak atau menampilkan output.
`for` berfungsi sebagai perulangan, yang merupakan proses mengulang-ulang eksekusi blok kode di dalam perintah `for` tanpa henti selama kondisi yang diperintahkan terpenuhi, dalam blok kode `for` juga biasanya terdapat variabel penanda kapan perulangan akan diberhentikan.
`temp` atau yang bisa kita sebut juga sebagai temporary variable adalah tempat penyimpanan sementara tanpa harus menyimpan input terbaru ke dalam suatu variabel yang dideklarasi.

Yang dilakukan program di dalam gambar di atas adalah **Mencetak N Bilangan Pertama Dalam Deret Fibonacci** yang mana terlihat melalui kode dari **baris 11 hingga baris 16**.

`for j < N {` → Memeriksa apakah kondisi *value* dalam variabel j kurang dari variabel N atau tidak, jika kurang dari atau ya, maka operasi dalam perulangan for akan dijalankan

`fmt.Print(s1, " ")` → Mencetak nilai dari variabel s1 yang sebelumnya sudah dideklarasikan, yaitu 0. Permisalan output: 0

`temp = s1 + s2` → Menyimpan nilai operasi hitung tambah dari variabel s1 dan s2 yang sebelumnya telah dideklarasi dalam program, yaitu 0 dan 1, maka hasilnya 1. Sehingga, nilai yang akan disimpan secara sementara dalam temp adalah 1

`s1 = s2` → Deklarasi nilai atau value variabel s1 adalah sama dengan nilai s2 yang sebelumnya dideklarasikan sebagai nilai terbaru dari variabel s1. Permisalan nilai terbaru s1: 1

`s2 = temp` → Deklarasi nilai atau value variabel s2 adalah sama dengan nilai temp yang sebelumnya disimpan sebagai nilai terbaru dari s2. Permisalan nilai terbaru s2: 1

`j = j + 1` → Menyimpan nilai terbaru variabel j dari operasi tambah variabel j dengan nilai 1 sebagai nilai terbaru dari variabel j untuk perulangan selanjutnya.

Runtutan Eksekusi:

1. Menunggu user melakukan input ke dalam variabel N yang akan dibaca lalu dimasukkan ke dalam variabel N
2. Memeriksa kondisi nilai variabel N sesuai dengan kondisi perulangan atau tidak, jika ya, maka akan dilanjutkan ke dalam operasi perulangan
3. Mencetak output nilai dari variabel s1 + ” ” (space) jika input N sesuai dengan ketentuan
4. Jika kondisi tidak sesuai dengan kondisi dalam perulangan sedari awal, maka operasi perulangan akan dihiarukan dan program berhenti

- Output dari perulangan ini akan selalu bertambah karena setiap variabel akan terus diupdate nilainya dan perulangan akan terhenti saat jumlah nilai j sudah lebih 1 nilai dari variabel N

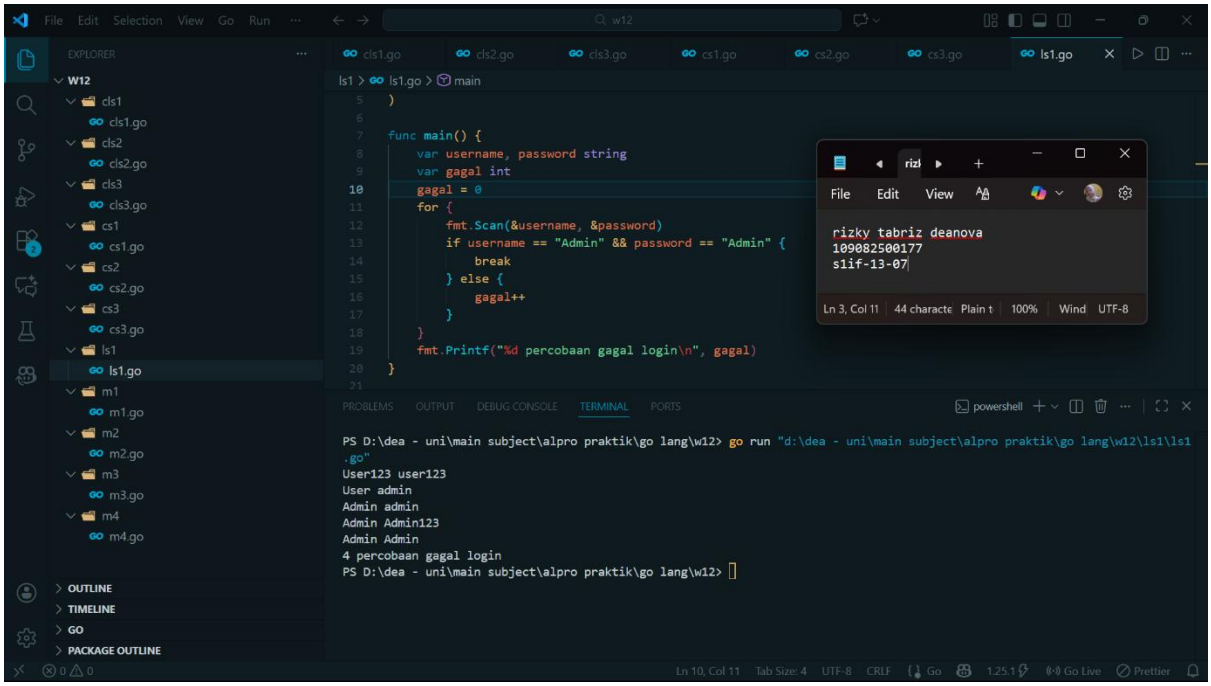
TUGAS

1. Tugas 1

Source code

```
package main
import (
    "fmt"
)
func main() {
    var username, password string
    var gagal int
    gagal = 0
    for {
        fmt.Scan(&username, &password)
        if username == "Admin" && password == "Admin" {
            break
        } else {
            gagal++
        }
    }
    fmt.Printf("%d percobaan gagal login\n", gagal)
}
```

Screenshoot program



Deskripsi program

`package main` sebagai penanda bahwa program ini merupakan program *executable* atau program yang dapat dieksekusi (dijalankan).

`import ("fmt")` yang digunakan untuk memasukkan library `"fmt"` untuk dipakai menjalankan program nantinya.

`func main () {}` yang berperan sebagai penanda tempat program mulai berjalan, seluruh kode ataupun instruksi yang terdapat di dalam kurung `{}` setelah `func main ()` nantinya akan dieksekusi atau dijalankan secara sistematis (berurutan).

`var` dalam program adalah sebagai wadah utama untuk penyimpanan data sebelum nanti dibagi lagi sesuai tipe data.

`fmt.Scan` yang dalam bahasa pemrograman Go digunakan untuk menjeda hasil output sejenak guna membaca input user, yang kemudian disimpan di dalam variabel.

`fmt.Printf` yang dalam bahasa pemrograman Go digunakan sebagai instruksi untuk mencetak atau menampilkan output yang terformat.

`for` berfungsi sebagai perulangan, yang merupakan proses mengulang-ulang eksekusi blok kode di dalam perintah `for` tanpa henti selama kondisi yang diperintahkan terpenuhi, dalam blok kode `for` juga biasanya terdapat variabel penanda kapan perulangan akan diberhentikan.

`if` berfungsi sebagai pengecekan kesesuaian kondisi suatu variabel dengan kondisi di dalam baris `if`.

`break` adalah *command* untuk menghentikan operasi `if` jika suatu variabel yang tercantum dalam operasi tersebut sesuai dengan kondisi di dalam operasi yang diperiksa.

`else` adalah *command* di mana jika suatu kondisi variabel tidak sesuai dengan kondisi di dalam operasi `if` maka operasi `else` akan dijalankan dan `if` akan dihiraukan

Yang dilakukan program di dalam gambar di atas adalah **Memeriksa Banyaknya Percobaan Kesesuaian Input Username & Password** yang mana terlihat melalui kode dari **baris 11 hingga baris 19**.

`for {` → Dimulainya perulangan dengan operasi dan juga kondisi yang terdapat di dalamnya

`fmt.Scan(&username, &password)` → Membaca input user untuk dimasukkan ke dalam variabel username dan password

`if username == "Admin" && password == "Admin" {` → Memeriksa kesesuaian input variabel username dan password dengan kondisi yang telah dicantumkan atau dideklarasikan di dalam operasi if

`break` → Menghentikan operasi for jikalau input yang diberikan user sesuai dengan kondisi di dalam operasi if

`} else {` → Operasi yang akan dieksekusi jika seluruh input tidak sesuai dengan kondisi if dan akan lanjut ke operasi yang ada di dalam kurung kurawal {}

`gagal++` → Menambah nilai atau *value* dalam variabel gagal sebanyak 1 jika input user tidak sesuai dengan kondisi if

`fmt.Printf("%d percobaan gagal login\n", gagal)` → Mencetak jumlah percobaan login dengan output terformat, yaitu integer yang mana tersimpan di dalam variabel gagal

Runtutan Eksekusi:

1. Deklarasi nilai variabel gagal sebanyak 0
2. Memulai operasi perulangan
3. Membaca serta menyimpan input user ke dalam variabel username dan password
4. Memeriksa kondisi input sesuai dengan kondisi operasi if atau tidak
5. Jika ya, maka akan lanjut ke operasi break yang mana menghentikan seluruh operasi perulangan dan akan langsung berpindah ke operasi print output jumlah percobaan login
6. Jika tidak, maka akan dilanjutkan ke operasi else yang mana akan menambah nilai atau value variabel gagal sebanyak 1 dan penambahan nilai ini akan menjadi nilai atau value variabel gagal yang terbaru
7. Mencetak output jumlah percobaan login dengan output terformat integer yang ada di dalam variabel gagal

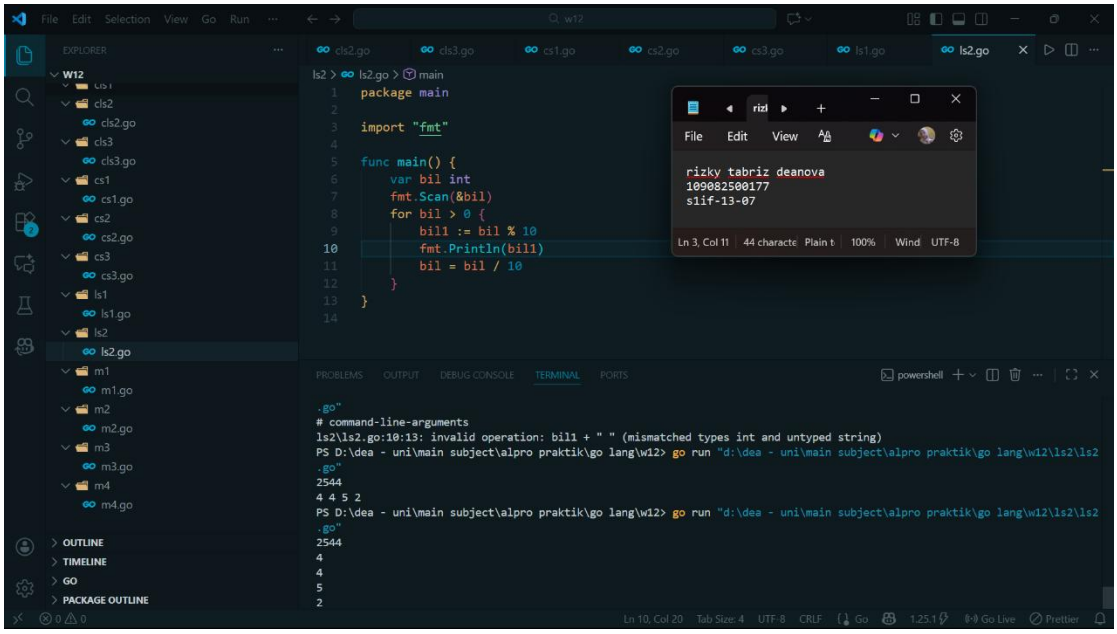
- Proses perulangan ini dimulai dari if yang mana jika tidak sesuai maka akan langsung loncat ke operasi else dan akan selalu berulang selama input variabel username dan password tidak sesuai, jika sesuai maka akan dihentikan dengan operasi break dan akan langsung beralih ke operasi pencetakan output terformat jumlah percobaan login

2. Tugas 2

Source code

```
package main
import "fmt"
func main() {
    var bil int
    fmt.Scan(&bil)
    for bil > 0 {
        bil1 := bil % 10
        fmt.Println(bil1)
        bil = bil / 10
    }
}
```

Screenshoot program



Deskripsi program

`package main` sebagai penanda bahwa program ini merupakan program *executable* atau program yang dapat dieksekusi (dijalankan).

`import ("fmt")` yang digunakan untuk memasukkan library `"fmt"` untuk dipakai menjalankan program nantinya.

`func main () {}` yang berperan sebagai penanda tempat program mulai berjalan, seluruh kode ataupun instruksi yang terdapat di dalam kurung `{}` setelah `func main ()` nantinya akan dieksekusi atau dijalankan secara sistematis (berurutan).

`var` dalam program adalah sebagai wadah utama untuk penyimpanan data sebelum nanti dibagi lagi sesuai tipe data.
`fmt.Scan` yang dalam bahasa pemrograman Go digunakan untuk menjeda hasil output sejenak guna membaca input user, yang kemudian disimpan di dalam variabel.
`fmt.Println` yang dalam bahasa pemrograman Go digunakan sebagai instruksi untuk mencetak atau menampilkan output di garis baru.
`for` berfungsi sebagai perulangan, yang merupakan proses mengulang-ulang eksekusi blok kode di dalam perintah `for` tanpa henti selama kondisi yang diperintahkan terpenuhi, dalam blok kode `for` juga biasanya terdapat variabel penanda kapan perulangan akan diberhentikan.

Yang dilakukan program di dalam gambar di atas adalah **Mencetak Deret Angka Dari Angka Terakhir** yang mana terlihat melalui kode dari **baris 8 hingga baris 11**.

`for bil > 0 {` → Memeriksa kondisi nilai variabel bil lebih dari 0 atau tidak, jika ya maka perulangan akan dimulai dan dilanjutkan ke operasi di dalam perulangan

`bil1 := bil % 10` → Mendeklarasikan variabel baru, yaitu bil1 sebagai wadah atau tempat dari hasil operasi modulo variabel bil dengan angka 10

`fmt.Println(bil1)` → Mencetak hasil atau nilai terbaru dari variabel bil1 di setiap baris baru

`bil = bil / 10` → Memeriksa nilai pembagian dari angka yang tersisa dalam variabel bil. Operasi ini juga berfungsi untuk memeriksa apakah nilai dalam variabel bil masih dapat digunakan untuk perulangan atau tidak, jika hasilnya 0 atau tidak dapat dibagi dengan 0, maka perulangan akan dihentikan

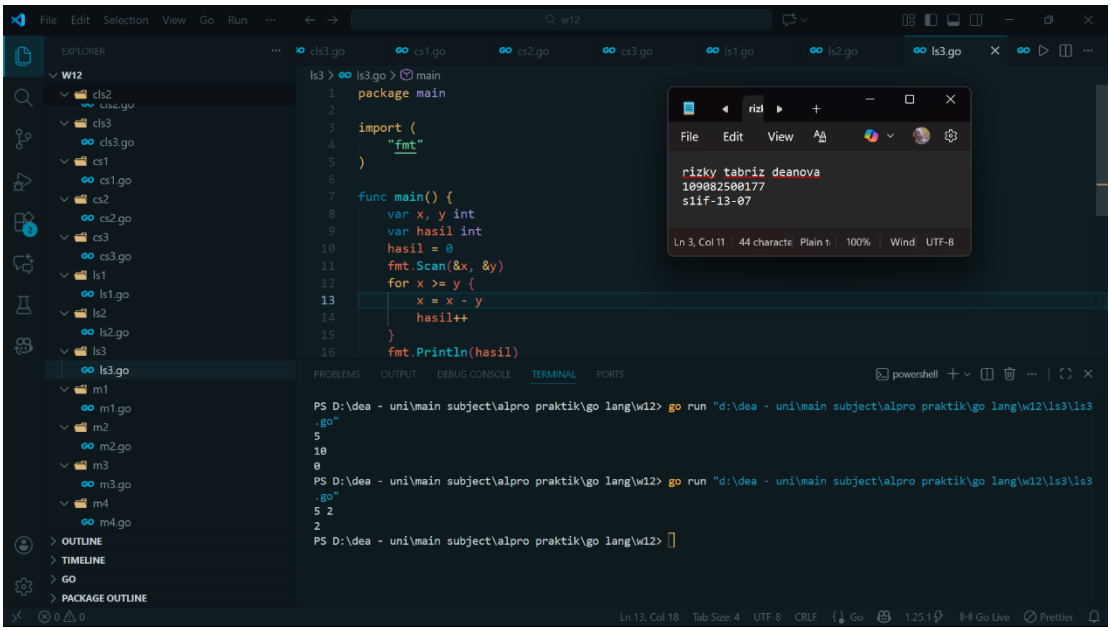
Runtutan Eksekusi:
1. Memeriksa kondisi nilai variabel bil sesuai dengan kondisi operasi for atau tidak
2. Jika ya, maka akan lanjut ke operasi di dalam for
3. Deklarasi variabel bil1 sebagai tempat dari hasil operasi modulo variabel bil dengan angka 10
4. Mencetak hasil dari operasi bil1
5. Memeriksa apakah nilai sisa dari variabel bil masih dapat digunakan atau sesuai dalam operasi perulangan tidak
6. Jika ya, maka operasi perulangan akan terus berulang atau dilakukan hingga nilai dari variabel bil tidak dapat digunakan dalam perulangan

3. Tugas 3

Source code

```
package main
import (
    "fmt"
)
func main() {
    var x, y int
    var hasil int
    hasil = 0
    fmt.Scan(&x, &y)
    for x >= y {
        x = x - y
        hasil++
    }
    fmt.Println(hasil)
}
```

Screenshoot program



Deskripsi program

`package main` sebagai penanda bahwa program ini merupakan program *executable* atau program yang dapat dieksekusi (dijalankan).

`import ("fmt")` yang digunakan untuk memasukkan library `"fmt"` untuk dipakai menjalankan program nantinya.

`func main () {}` yang berperan sebagai penanda tempat program mulai berjalan, seluruh kode ataupun instruksi yang terdapat di dalam kurung `{}` setelah `func main ()` nantinya akan dieksekusi atau dijalankan secara sistematis (berurutan).

`var` dalam program adalah sebagai wadah utama untuk penyimpanan data sebelum nanti dibagi lagi sesuai tipe data.

`fmt.Scan` yang dalam bahasa pemrograman Go digunakan untuk menjeda hasil output sejenak guna membaca input user, yang kemudian disimpan di dalam variabel.

`fmt.Println` yang dalam bahasa pemrograman Go digunakan sebagai instruksi untuk mencetak atau menampilkan output di garis baru.
`for` berfungsi sebagai perulangan, yang merupakan proses mengulang-ulang eksekusi blok kode di dalam perintah `for` tanpa henti selama kondisi yang diperintahkan terpenuhi, dalam blok kode `for` juga biasanya terdapat variabel penanda kapan perulangan akan diberhentikan.

Yang dilakukan program di dalam gambar di atas adalah **Mencetak Hasil Integer Division Dua Bilangan Tanpa Menggunakan Integer Division** yang mana terlihat melalui kode dari **baris 12 hingga baris 14**.

`for x >= y {` → Memeriksa kondisi nilai variabel x lebih dari atau sama dengan variabel y atau tidak, jika ya maka perulangan akan dimulai dan dilanjutkan ke operasi di dalam perulangan

`x = x - y` → Melakukan operasi pengurangan antara variabel x dengan variabel y, setelahnya hasil dari pengurangan tersebut akan menjadi nilai terbaru dari variabel x (hasil akan disimpan dalam variabel x)

`hasil++` → Setiap operasi perulangan yang dilakukan, nilai variabel hasil akan bertambah 1 yang totalnya akan dijadikan sebagai hasil dari pembagian 2 bilangan tanpa harus menggunakan *integer division*

Runtutan Eksekusi:
1. Memeriksa kondisi nilai variabel x lebih dari atau sama dengan variabel y atau tidak, jika ya maka perulangan akan dimulai dan dilanjutkan ke operasi di dalam perulangan
2. Nilai dari variabel x akan dikurangi dengan nilai variabel y, yang nantinya hasil dari operasi pengurangan ini akan menjadi nilai terbaru dari variabel x
3. Nilai dari variabel hasil akan bertambah 1 setiap perulangan dilakukan
4. Mencetak output dari variabel hasil sebagai nilai dari pembagian 2 bilangan tanpa menggunakan *integer division* di garis baru

- Nilai dari variabel hasil mewakilkan hasil pembagian 2 bilangan