**LAPORAN PRAKTIKUM ALGORITMA**

**DAN PEMROGRAMAN 1**

**MODUL 13**

**REPEAT-UNTIL**


**Disusun oleh:**

**ALIN KARISA HIZANNAH**

**109082500010**

**S1IF-13-07**

**Asisten Praktikum**

Adithana dharma putra

Apri pandu wicaksono

**PROGRAM STUDI S1 INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2025**

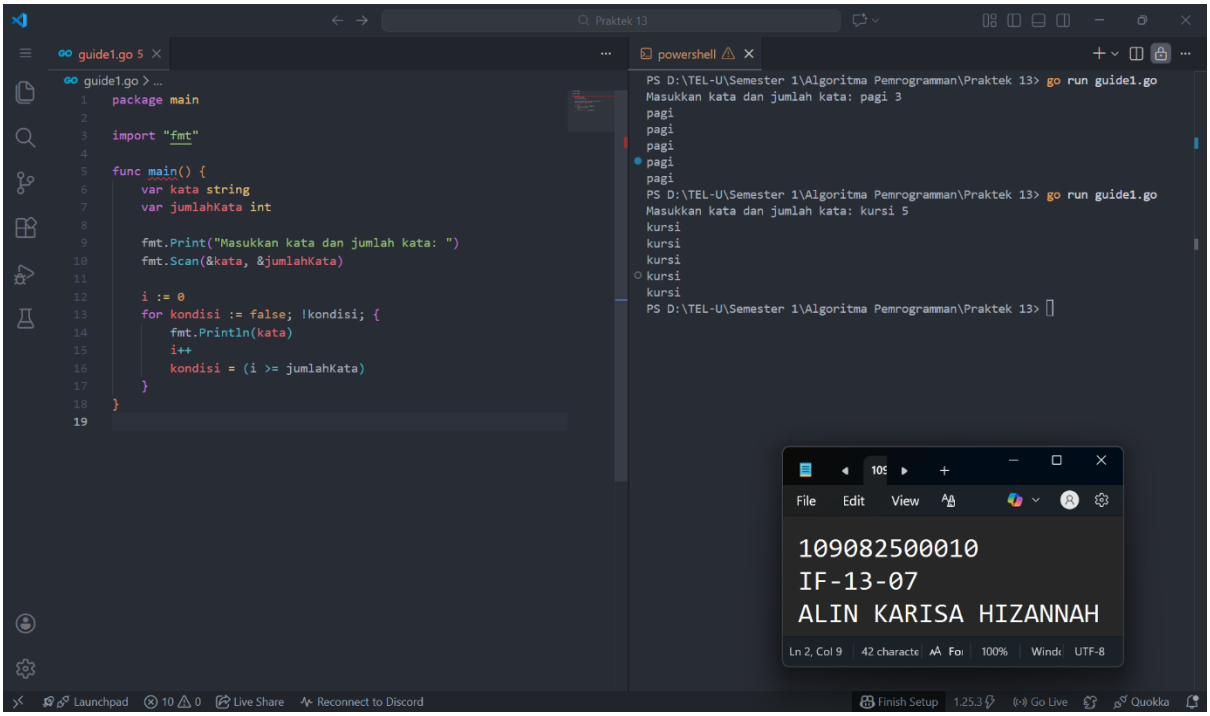
**LATIHAN KELAS – GUIDED**

1. **Guided 1**

**Source Code**

```
package main


import "fmt"


func main() {
    var kata string
    var jumlahKata int


    fmt.Print("Masukkan kata dan jumlah kata: ")
    fmt.Scan(&kata, &jumlahKata)


    i := 0
    for kondisi := false; !kondisi; {
        fmt.Println(kata)
        i++
        kondisi = (i >= jumlahKata)
    }
}
```

**Screenshoot program**




**Deskripsi program**

Program di atas merupakan implementasi perulangan repeat–until dalam bahasa pemrograman Go. Program ini bertujuan untuk menampilkan sebuah kata sebanyak jumlah tertentu sesuai dengan input yang diberikan oleh pengguna. Perulangan dibuat menggunakan struktur for dengan kondisi boolean, sehingga perulangan akan terus berjalan hingga kondisi berhenti terpenuhi.

Pada bagian awal program, dilakukan deklarasi dua variabel, yaitu kata bertipe string untuk menyimpan teks yang akan ditampilkan, dan jumlahKata bertipe int untuk menentukan berapa kali kata tersebut dicetak. Program kemudian menampilkan pesan “Masukkan kata dan jumlah kata:” sebagai petunjuk input, lalu membaca dua nilai sekaligus menggunakan fmt.Scan, yaitu kata dan jumlah pengulangan.

Selanjutnya, variabel penghitung i diinisialisasi dengan nilai nol. Variabel ini berfungsi untuk menghitung berapa kali kata telah ditampilkan. Perulangan dilakukan menggunakan struktur for kondisi := false; !kondisi; {}, yang merepresentasikan konsep repeat–until, di mana blok perulangan akan dijalankan minimal satu kali sebelum kondisi berhenti diperiksa.

Di dalam perulangan, program mencetak nilai variabel kata ke layar menggunakan fmt.Println. Setelah itu, nilai i ditambah satu setiap kali perulangan

berjalan. Kondisi berhenti ditentukan dengan pernyataan kondisi = (i >= jumlahKata), yang berarti perulangan akan berhenti ketika jumlah cetakan kata telah mencapai atau melebihi jumlah yang diminta pengguna.

Berdasarkan hasil eksekusi program pada terminal, terlihat bahwa ketika pengguna memasukkan input pagi 3, kata “pagi” ditampilkan sebanyak tiga kali. Begitu pula saat input kursi 5 diberikan, kata “kursi” ditampilkan sebanyak lima kali. Hal ini menunjukkan bahwa program telah berjalan sesuai dengan logika yang dirancang dan berhasil mengimplementasikan perulangan repeat–until dengan benar.

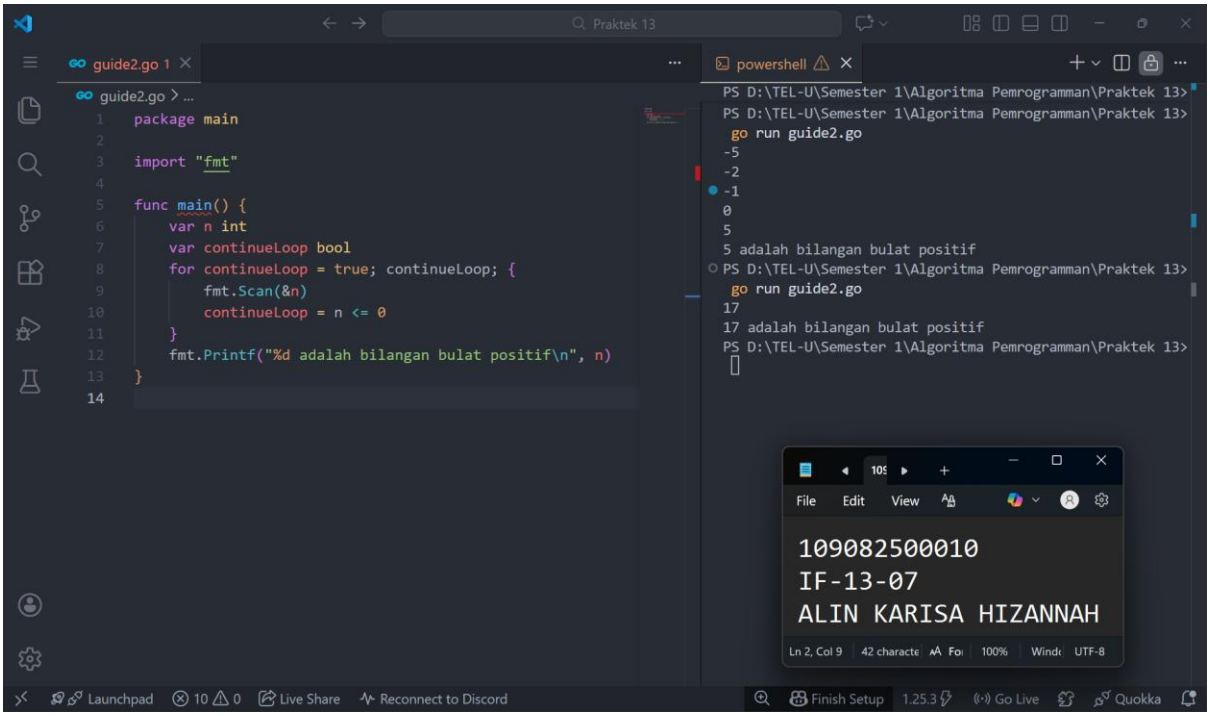
Program ini menunjukkan penggunaan dasar struktur perulangan, variabel penghitung, serta mekanisme input dan output pada bahasa Go. Dengan pendekatan ini, pengguna dapat memahami bagaimana sebuah perulangan dikendalikan menggunakan kondisi boolean dan bagaimana perulangan dapat dijalankan minimal satu kali sebelum berhenti.

## 2. Guided 2

### Source Code

```
package main


import "fmt"


func main() {
    var n int
    var continueLoop bool
    for continueLoop = true; continueLoop; {
        fmt.Scan(&n)
        continueLoop = n <= 0
    }
    fmt.Printf("%d adalah bilangan bulat positif\n", n)
}
```

### Screenshoot program




**Deskripsi program**

Proses yang ditunjukkan memperlihatkan mekanisme validasi input untuk memastikan bahwa nilai yang dimasukkan oleh pengguna merupakan bilangan bulat positif. Sistem akan terus menerima masukan selama nilai yang diberikan masih bernilai nol atau negatif, sehingga hanya nilai yang memenuhi kriteria tertentu yang dapat diproses lebih lanjut.

Pada tahap awal, sebuah variabel bilangan bulat digunakan untuk menampung nilai masukan dari pengguna, sedangkan variabel bertipe boolean dimanfaatkan sebagai pengendali perulangan. Selama kondisi pengendali bernilai benar, proses pembacaan input akan terus dilakukan. Pendekatan ini memungkinkan pemeriksaan kondisi dilakukan secara berulang hingga syarat yang ditentukan terpenuhi.

Setiap kali pengguna memasukkan sebuah nilai, kondisi perulangan akan diperbarui berdasarkan hasil evaluasi apakah nilai tersebut kurang dari atau sama dengan nol. Jika kondisi tersebut terpenuhi, proses akan diulang dan pengguna diminta untuk memasukkan nilai kembali. Sebaliknya, ketika nilai yang dimasukkan lebih besar dari nol, perulangan akan dihentikan secara otomatis.

Setelah kondisi terpenuhi, hasil akhir ditampilkan dalam bentuk pernyataan bahwa nilai yang dimasukkan merupakan bilangan bulat positif. Tampilan ini hanya muncul setelah proses validasi selesai, sehingga dapat dipastikan bahwa nilai yang ditampilkan telah sesuai dengan ketentuan yang ditetapkan.

Berdasarkan hasil eksekusi yang ditunjukkan, masukan berupa bilangan negatif dan nol tidak langsung menghasilkan keluaran akhir. Proses baru berhenti ketika pengguna memasukkan bilangan positif, seperti 5 atau 17, dan sistem kemudian menampilkan pernyataan yang sesuai. Hal ini menunjukkan bahwa mekanisme perulangan dan validasi input telah berjalan dengan baik dan konsisten.

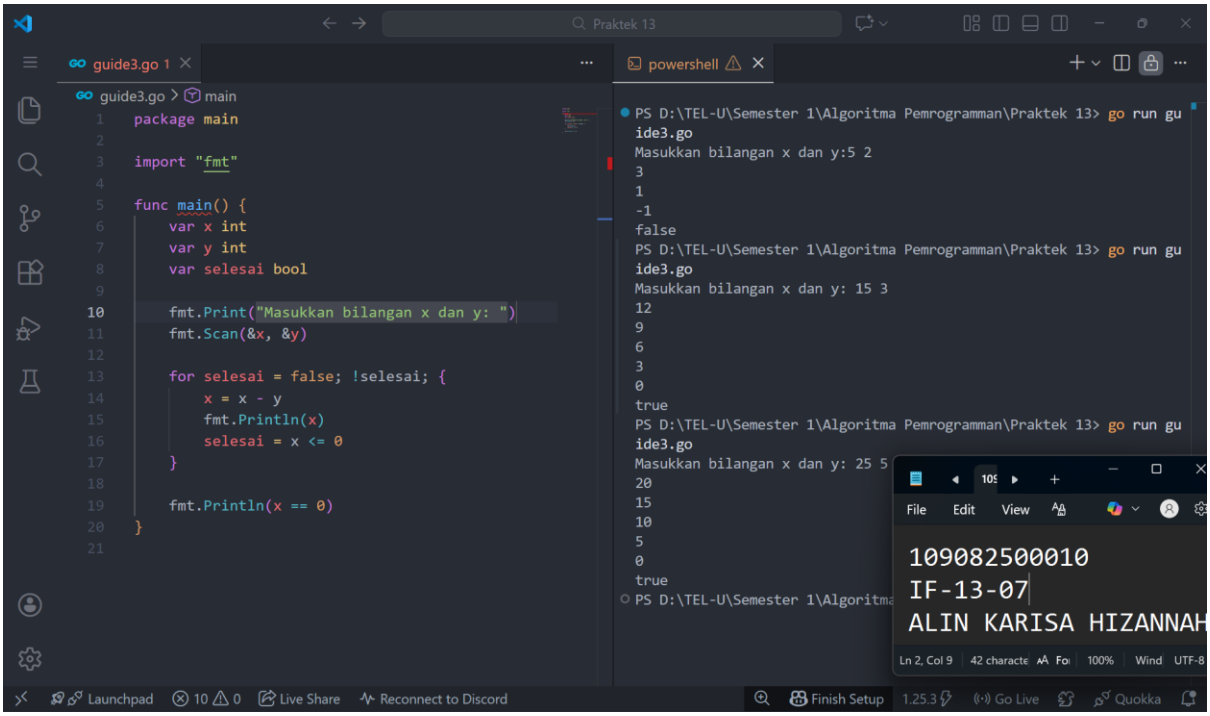
Pendekatan tersebut efektif untuk mencegah kesalahan input dan memastikan bahwa data yang diproses telah sesuai dengan kriteria yang diharapkan. Dengan demikian, validasi input dapat dilakukan secara sistematis dan berulang tanpa memerlukan pemeriksaan manual tambahan.

3. **Guided 3**

**Source Code**

```
package main


import "fmt"


func main() {
    var x int
    var y int
    var selesai bool


    fmt.Print("Masukkan bilangan x dan y: ")
    fmt.Scan(&x, &y)


    for selesai = false; !selesai; {
        x = x - y
        fmt.Println(x)
        selesai = x <= 0
    }


    fmt.Println(x == 0)
}
```

**Screenshoot program**




**Deskripsi program**

Alur yang ditunjukkan memperlihatkan proses pengurangan berulang antara dua bilangan bulat yang dimasukkan oleh pengguna. Pengguna diminta memasukkan dua nilai, yaitu bilangan x sebagai nilai awal dan bilangan y sebagai nilai pengurang. Proses pengurangan dilakukan secara bertahap hingga nilai x mencapai nol atau bernilai negatif.

Pada tahap awal, nilai x dan y dibaca dari masukan pengguna. Selain itu, digunakan sebuah variabel bertipe boolean sebagai penanda untuk menentukan apakah proses pengulangan masih harus dijalankan atau dihentikan. Pendekatan ini memungkinkan pengendalian perulangan dilakukan secara dinamis berdasarkan hasil operasi yang sedang berlangsung.

Setiap kali perulangan dijalankan, nilai x dikurangi dengan nilai y, kemudian hasil pengurangan tersebut langsung ditampilkan ke layar. Proses ini berlangsung secara berulang sehingga pengguna dapat melihat perubahan nilai x secara bertahap. Setelah setiap pengurangan, kondisi penghentian diperiksa dengan mengevaluasi apakah nilai x sudah kurang dari atau sama dengan nol.

Ketika kondisi tersebut terpenuhi, proses pengulangan dihentikan dan sistem menampilkan hasil evaluasi berupa nilai logika. Nilai logika ini menunjukkan apakah

nilai akhir x sama dengan nol atau tidak. Jika hasilnya bernilai true, berarti nilai x habis dibagi oleh y tanpa sisa. Sebaliknya, jika bernilai false, berarti nilai x menjadi negatif sebelum mencapai nol.

Hasil eksekusi yang ditampilkan menunjukkan bahwa untuk beberapa kombinasi nilai masukan, nilai x dapat mencapai nol secara tepat, sedangkan pada kombinasi lain nilai x berhenti pada bilangan negatif. Hal ini memperlihatkan bagaimana proses pengurangan berulang dapat digunakan untuk mengevaluasi keterbagian suatu bilangan terhadap bilangan lain secara logis.

Pendekatan ini memberikan pemahaman mengenai penggunaan perulangan berbasis kondisi boolean serta penerapan operasi aritmetika sederhana dalam proses iteratif. Dengan mekanisme tersebut, alur perhitungan dapat diamati secara bertahap hingga kondisi penghentian terpenuhi.

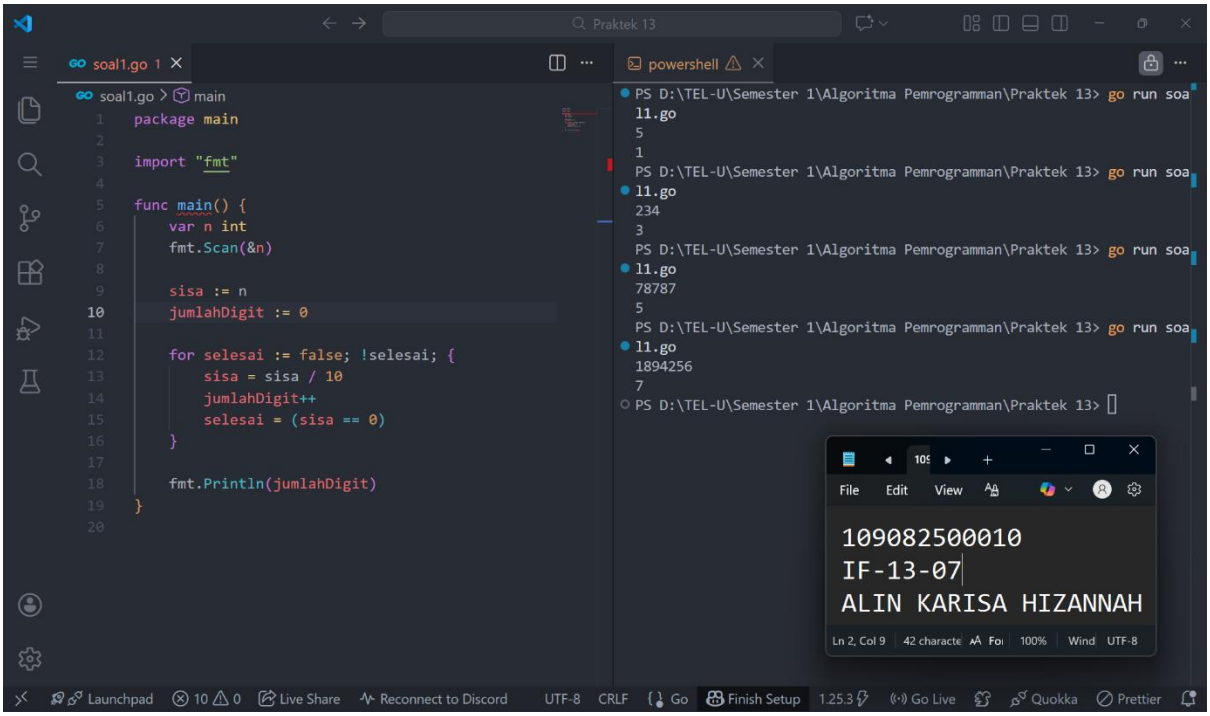
**TUGAS**

1. **Tugas 1**

**Source code**

```
package main


import "fmt"


func main() {
    var n int
    fmt.Scan(&n)


    sisa := n
    jumlahDigit := 0


    for selesai := false; !selesai; {
        sisa = sisa / 10
        jumlahDigit++
        selesai = (sisa == 0)
    }


    fmt.Println(jumlahDigit)
}
```

**Screenshoot program**




**Deskripsi program**

Bagian ini menjelaskan proses untuk menentukan jumlah digit dari sebuah bilangan bulat yang dimasukkan oleh pengguna. Pengguna diminta memasukkan satu nilai bilangan, kemudian sistem akan mengolah nilai tersebut untuk mengetahui berapa banyak digit yang menyusunnya.

Bilangan yang dimasukkan disimpan terlebih dahulu, lalu diproses dengan cara dibagi sepuluh secara berulang. Setiap kali pembagian dilakukan, satu digit dianggap telah dihitung. Proses ini terus berlangsung sampai nilai hasil pembagian menjadi nol, yang menandakan bahwa seluruh digit pada bilangan tersebut telah selesai diperiksa.

Selama proses perhitungan berlangsung, sebuah variabel penghitung digunakan untuk mencatat jumlah pembagian yang terjadi. Setiap penambahan nilai pada variabel ini menunjukkan bertambahnya satu digit. Dengan pendekatan tersebut, jumlah digit dapat diketahui tanpa perlu mengubah bilangan menjadi bentuk teks.

Hasil yang ditampilkan menunjukkan jumlah digit sesuai dengan bilangan yang dimasukkan. Sebagai contoh, ketika pengguna memasukkan angka 234, hasil yang

diperoleh adalah 3, sedangkan pada angka 78787 hasil yang ditampilkan adalah 5. Hal ini menunjukkan bahwa perhitungan digit dilakukan secara tepat.

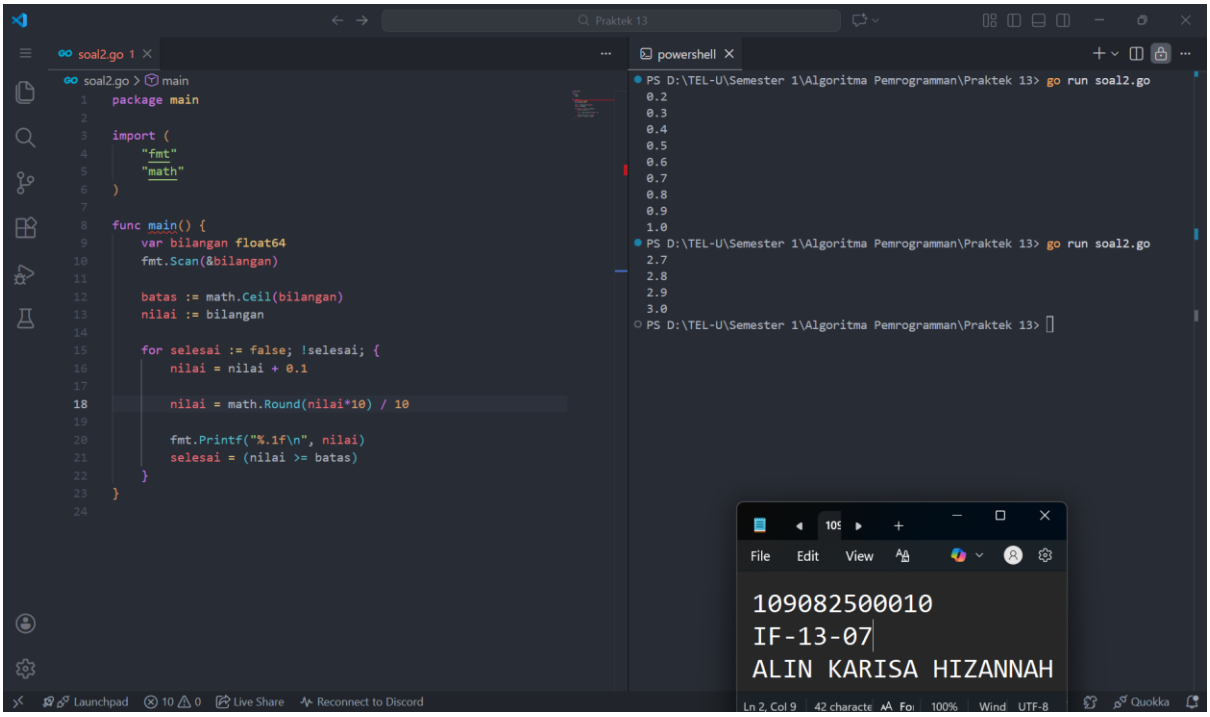
Pendekatan ini memudahkan pemahaman mengenai bagaimana operasi pembagian dan perulangan dapat dimanfaatkan untuk menyelesaikan permasalahan sederhana. Selain itu, latihan ini juga membantu memperkuat pemahaman tentang penggunaan variabel penghitung dan kondisi penghentian dalam proses pengulangan.

2. **Tugas 2**

**Source code**

```
package main


import (
    "fmt"
    "math"
)


func main() {
    var bilangan float64
    fmt.Scan(&bilangan)


    batas := math.Ceil(bilangan)
    nilai := bilangan


    for selesai := false; !selesai; {
        nilai = nilai + 0.1


        nilai = math.Round(nilai*10) / 10


        fmt.Printf("%.1f\n", nilai)
        selesai = (nilai >= batas)
    }
}
```

**Screenshoot program**




**Deskripsi program**

Kode tersebut digunakan untuk menampilkan bilangan desimal secara bertahap hingga mencapai bilangan bulat terdekat ke atas. Nilai awal dimasukkan oleh pengguna, kemudian angka tersebut dinaikkan sedikit demi sedikit dengan selisih yang sama pada setiap perulangan.

Setiap hasil penambahan ditampilkan ke layar sehingga pengguna dapat melihat perubahan nilai dari angka awal sampai mencapai batas pembulatan. Perulangan dihentikan ketika nilai yang ditampilkan sudah mencapai bilangan bulat tujuan, sehingga hasil akhir tidak melewati batas yang diinginkan.

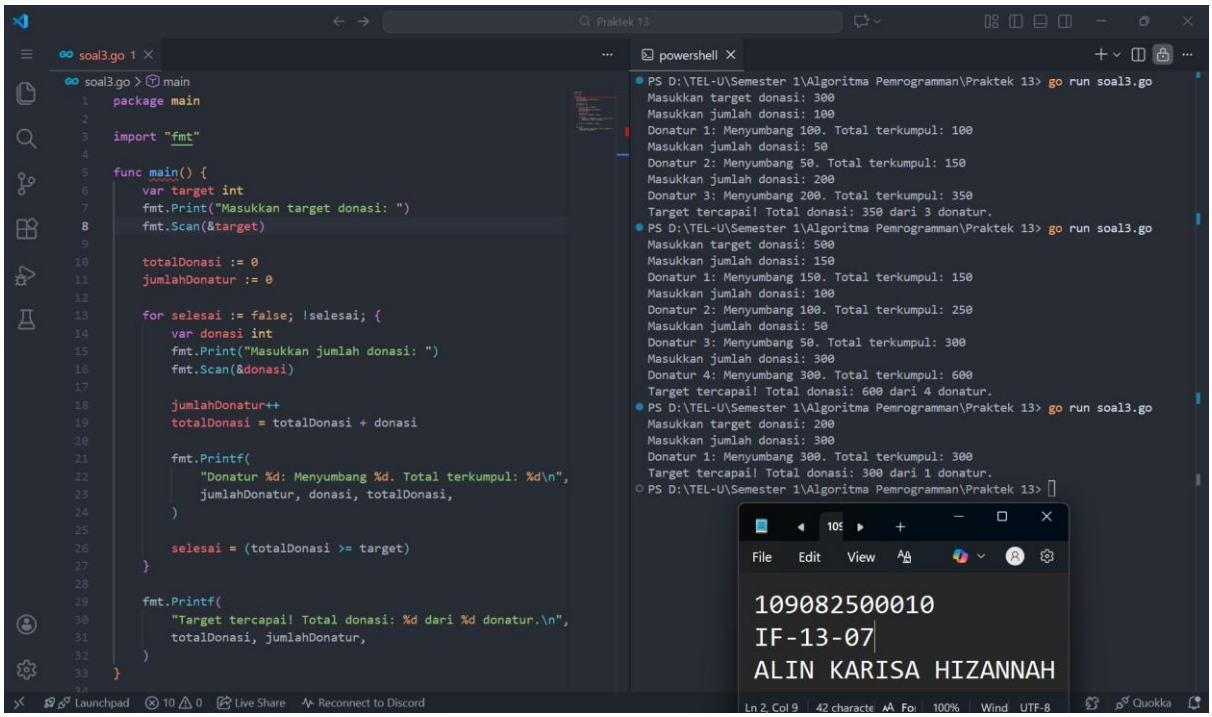
3. **Tugas 3**

**Source code**

```
package main


import "fmt"


func main() {
    var target int
    fmt.Print("Masukkan target donasi: ")
    fmt.Scan(&target)


    totalDonasi := 0
    jumlahDonatur := 0


    for selesai := false; !selesai; {
        var donasi int
        fmt.Print("Masukkan jumlah donasi: ")
        fmt.Scan(&donasi)


        jumlahDonatur++
        totalDonasi = totalDonasi + donasi


        fmt.Printf(
            "Donatur %d: Menyumbang %d. Total
terkumpul: %d\n",
            jumlahDonatur, donasi, totalDonasi,
```

```
            )


            selesai = (totalDonasi >= target)
        }


        fmt.Printf(
            "Target tercapai! Total donasi: %d dari %d
    donatur.\n",
            totalDonasi, jumlahDonatur,
        )
    }
```

**Screenshoot program**




**Deskripsi program**

Pengguna diminta memasukkan target donasi yang ingin dicapai. Setelah nilai target dimasukkan, sistem kemudian meminta jumlah donasi dari donatur pertama. Nilai donasi tersebut langsung dijumlahkan ke total donasi dan ditampilkan bersama informasi nomor donatur serta total donasi sementara.

Setelah donasi pertama diproses, sistem kembali meminta jumlah donasi berikutnya. Setiap donasi yang dimasukkan akan meningkatkan total donasi dan jumlah donatur. Informasi yang ditampilkan selalu mencakup nomor donatur, jumlah donasi yang diberikan, serta total donasi yang telah terkumpul hingga saat itu.

Proses input donasi terus berlanjut selama total donasi yang terkumpul masih belum mencapai target yang ditentukan. Pada gambar terlihat bahwa beberapa kali pengguna memasukkan nilai donasi yang berbeda-beda, dan setiap nilai tersebut langsung memengaruhi total donasi yang ditampilkan.

Ketika total donasi sudah sama dengan atau melebihi target, sistem menampilkan pesan bahwa target telah tercapai. Pesan tersebut juga menyertakan total akhir donasi yang berhasil dikumpulkan serta jumlah donatur yang terlibat dalam proses pengumpulan donasi tersebut.

Hasil yang ditampilkan pada terminal menunjukkan bahwa target donasi dapat tercapai dengan jumlah donatur yang bervariasi. Pada contoh yang terlihat, target donasi tertentu dapat tercapai hanya dengan satu donatur, sementara pada contoh lain diperlukan beberapa donatur hingga total donasi melebihi target.