**LAPORAN PRAKTIKUM**

**Algoritma Pemrograman**

**MODUL 13**

**REPEAT-UNTIL**


**Disusun oleh:**

**EDWARD ABIMAS SURYA HATTA**

**109082500171**

**S1IF-13-04**

**PROGRAM STUDI S1 INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2025**

**LATIHAN KELAS – GUIDED**

1. **Guided 1**

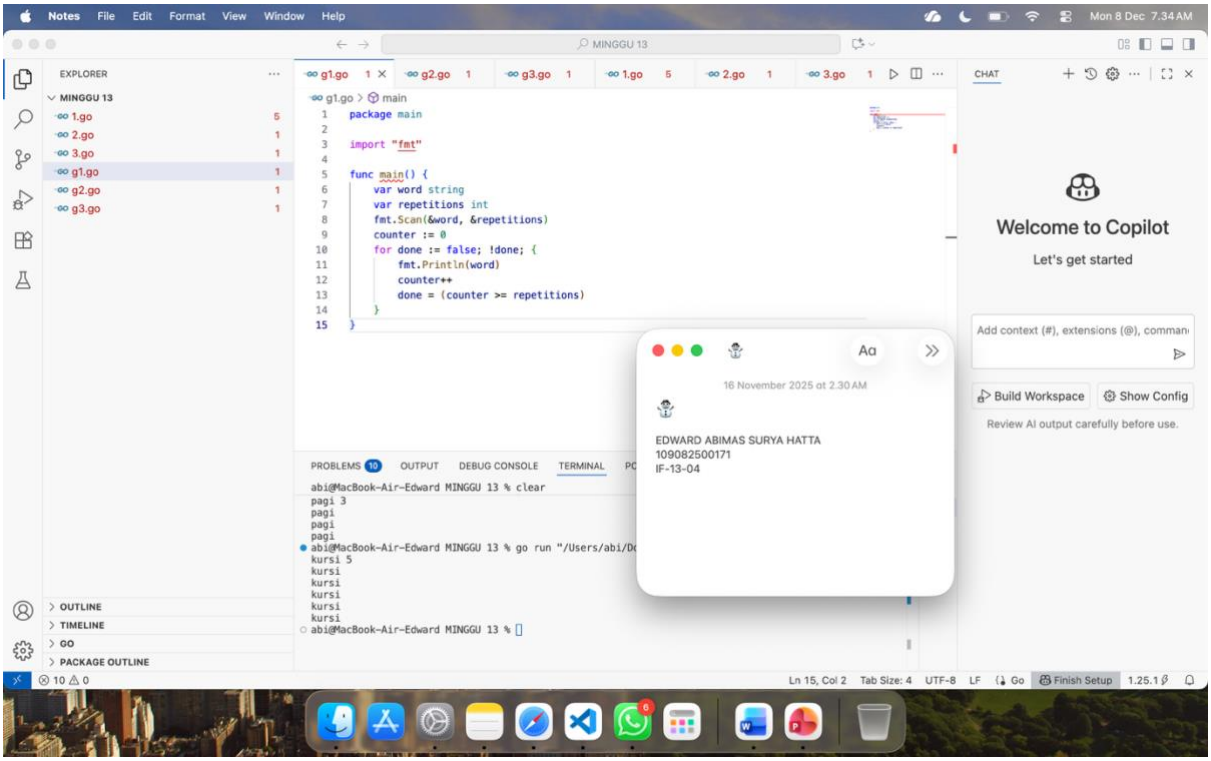
**Source Code**

```
package main


import "fmt"


func main() {
    var word string
    var repetitions int
    fmt.Scan(&word, &repetitions)
    counter := 0
    for done := false; !done; {
        fmt.Println(word)
        counter++
        done = (counter >= repetitions)
    }
}
```

**Screenshoot program**




**Deskripsi program**

Program ini dimulai dengan meminta komputer menyiapkan dua wadah (variabel), satu untuk kata yang ingin diulang (word) bertipe teks, dan satu lagi untuk berapa kali pengulangannya (repetitions) bertipe angka bulat. Setelah pengguna mengetikkan kata dan angkanya, program menyiapkan sebuah penghitung bernama counter yang dimulai dari nol.

Masuk ke bagian perulangan (loop), ini adalah inti dari konsep *Repeat-Until*. Go tidak punya perintah repeat, jadi kita mengakalinya dengan for done := false; !done;. Artinya, kita buat tanda bernama done yang awalnya false (belum selesai). Komputer disuruh bekerja terus selama done itu TIDAK benar (masih false). Karena di awal sudah kita set false, maka komputer pasti masuk ke dalam perulangan minimal satu kali. Di dalam, komputer mencetak kata tersebut ke layar, lalu menaikkan nilai counter sebanyak satu.

Setelah mencetak, komputer melakukan pengecekan: "Apakah counter saya sudah sama atau lebih besar dari jumlah repetitions yang diminta user?". Hasil pengecekan ini (Ya/Tidak) disimpan ke variabel done. Jika counter masih kurang, donetetap false, dan perulangan berlanjut. Tapi begitu counter sudah mencapai target, done berubah

jadi true. Saat kembali ke atas perulangan, syarat !done (tidak selesai) menjadi salah, sehingga komputer berhenti mengulang.

2. **Guided 2**

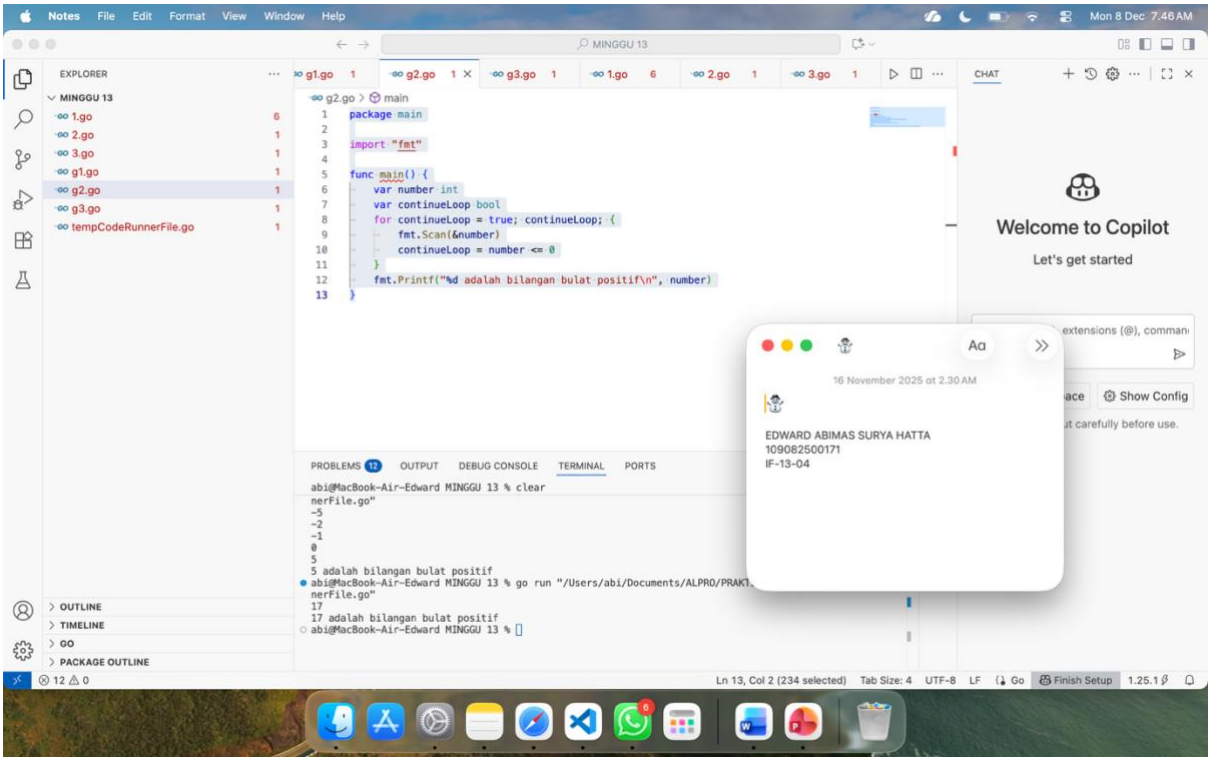
**Source Code**

```
package main


import "fmt"


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

**Screenshoot program**




**Deskripsi program**

Program ini bertujuan memaksa pengguna memasukkan angka positif. Pertama, komputer menyiapkan variabel number untuk menyimpan angka dan continueLoop (lanjutkan perulangan) sebagai penanda. Penanda continueLoop langsung diset trueagar pintu gerbang perulangan terbuka lebar untuk pertama kalinya.

Di dalam perulangan, komputer diam menunggu pengguna mengetik angka (fmt.Scan). Setelah angka masuk, komputer langsung mengecek kondisi logika: "Apakah angka ini kurang dari atau sama dengan nol?". Jika pengguna memasukkan angka negatif atau nol (misal -5), jawaban logikanya adalah "Ya (True)". Maka, continueLoop bernilai true, dan komputer akan mengulang lagi dari awal, meminta angka lagi. Ini akan terus terjadi sampai pengguna menyerah dan memasukkan angka positif (misal 10). Saat itu terjadi, pertanyaan "Apakah 10 <= 0?" jawabannya "Tidak (False)". Variabel continueLoopmenjadi false, perulangan berhenti, dan barulah komputer mencetak pesan bahwa itu adalah bilangan bulat positif.

3. **Guided 3**

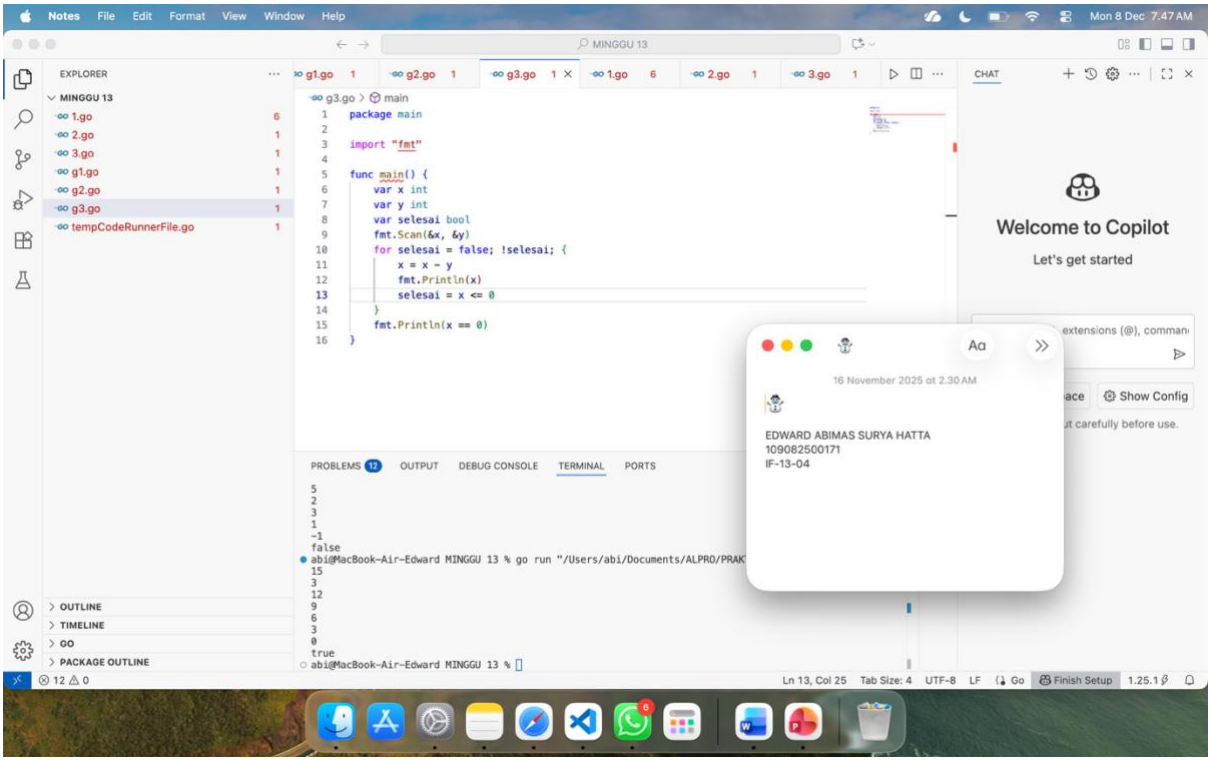
**Source Code**

```
package main


import "fmt"


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

Di sini komputer diminta mengecek apakah x adalah kelipatan y tanpa memakai pembagian, tapi dengan pengurangan terus-menerus. Setelah x dan y diinput, kita siapkan penanda selesai yang awalnya false. Perulangan pun dimulai karena syaratnya "lakukan selama TIDAK selesai".

Di dalam perulangan, nilai x dikurangi y (misal 10 dikurang 2 jadi 8). Hasil pengurangan itu dicetak. Kemudian komputer mengecek kondisi berhenti: "Apakah x sekarang sudah kurang dari atau sama dengan nol?". Selama x masih positif (masih ada sisa), selesai tetap false dan pengurangan berlanjut. Jika x akhirnya menjadi 0 (habis dibagi) atau minus (tidak habis dibagi), maka selesai berubah jadi true, dan perulangan berhenti. Terakhir, komputer membandingkan: kalau x akhirnya tepat 0, berarti benar itu kelipatan (true), kalau minus berarti bukan (false).

## TUGAS

1. **Tugas 1**

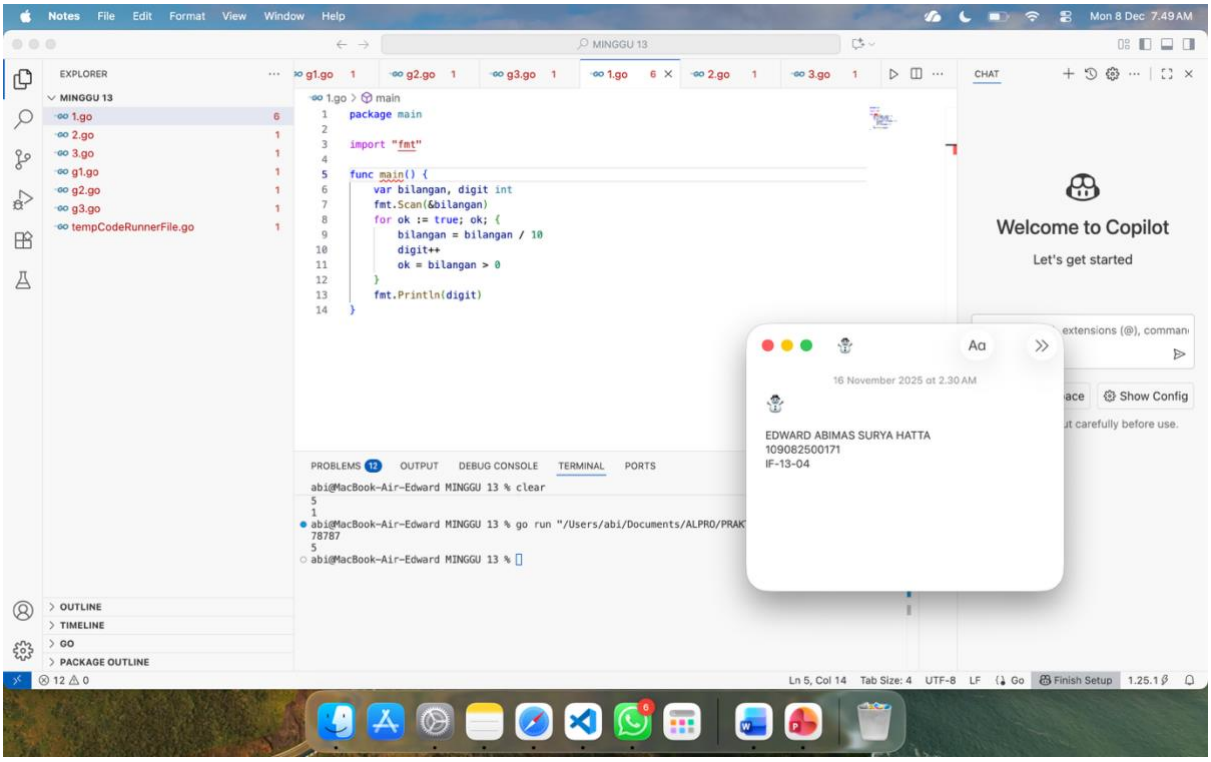
   **Source code**

```
package main
```

```
import "fmt"


func main() {
    var bilangan, digit int
    fmt.Scan(&bilangan)
    for ok := true; ok; {
        bilangan = bilangan / 10
        digit++
        ok = bilangan > 0
    }
    fmt.Println(digit)
}
```

**Screenshoot program**




**Deskripsi program**

Logika program ini adalah "memotong" angka paling belakang satu per satu sampai habis. Kita punya variabel bilangan(input user) dan digit (penghitung jumlah potongan) yang mulai dari 0. Perulangan disiapkan dengan penanda ok yang diset true agar mesin masuk ke loop minimal sekali.

Di dalam loop, perintah bilangan = bilangan / 10 dijalankan. Karena ini bilangan bulat, pembagian dengan 10 akan membuang angka terakhir (contoh: 123 dibagi 10 jadi 12, angka 3 hilang). Setiap kali satu angka "dibuang", variabel digit ditambah satu. Selanjutnya, komputer mengecek: "Apakah bilangan masih lebih besar dari 0?". Jika sisa 12, kan masih > 0, jadi oktetap true dan loop berlanjut memotong lagi jadi 1. Lanjut lagi potong jadi 0. Saat bilangan sudah 0, syarat bilangan > 0menjadi false. Variabel ok jadi false, perulangan berhenti, dan komputer menampilkan berapa kali ia memotong tadi (jumlah digitnya).

## 2. Tugas 2

**Source code**

```
package main


import (
    "fmt"
    "math"
)


func main() {
    var x float64
    fmt.Scan(&x)
    target := math.Ceil(x)
    for ok := true; ok; {
        x += 0.1
```

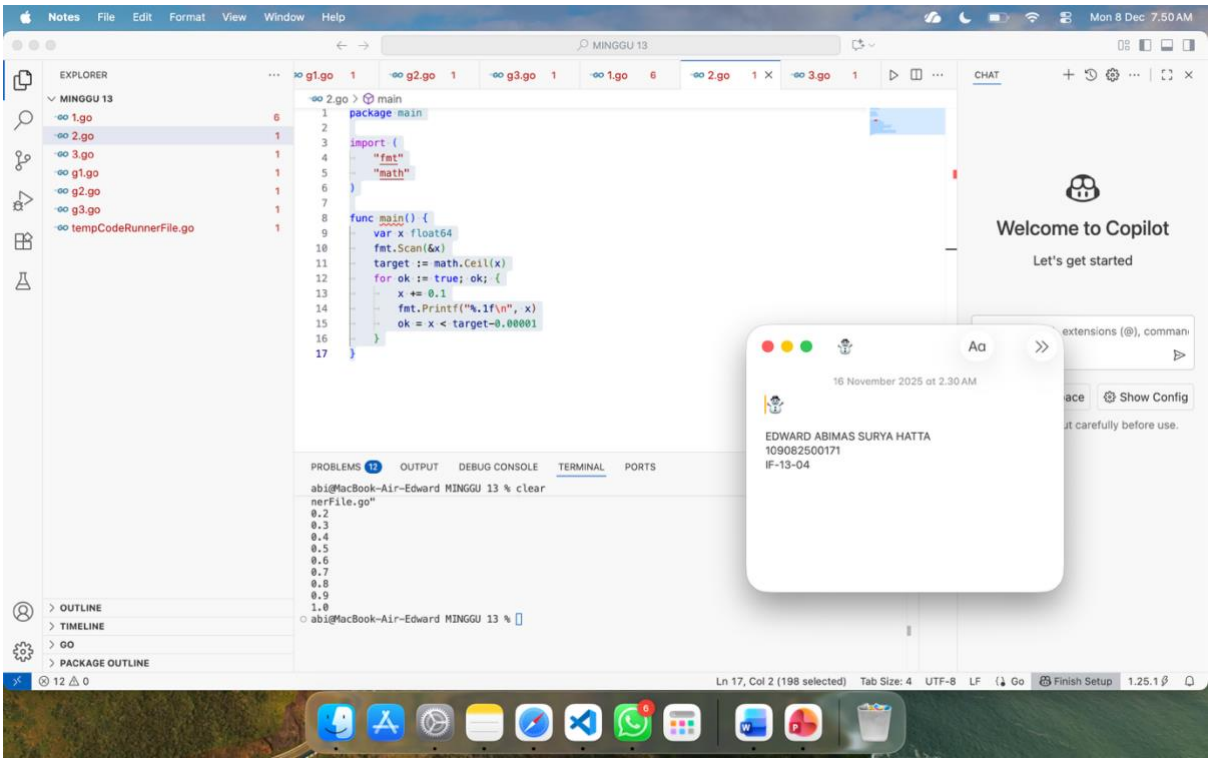
```
                fmt.Printf("%.1f\n", x)

                ok = x < target-0.00001

            }

        }
```

**Screenshoot program**




**Deskripsi program**

Program ini agak rumit karena berurusan dengan angka koma (float) yang sering tidak presisi di komputer. Pertama, xdibaca. Lalu komputer menghitung target pembulatan ke atas menggunakan math.Ceil (misal input 2.3, targetnya 3.0). Perulangan dimulai dengan penanda ok bernilai true.

Setiap putaran, x ditambah 0.1. Lalu x dicetak dengan format satu angka di belakang koma. Bagian kuncinya ada di syarat berhenti. Kita tidak bisa sekadar bilang "berhenti kalau x == target" karena di dunia komputer, 2.9 + 0.1 bisa jadi 2.99999999, bukan 3.0 pas. Jadi, syarat berhentinya kita balik: "Apakah x masih kurang dari target?". Untuk amannya, kita kurangi target dengan angka sangat kecil (0.00001) sebagai toleransi

error. Jadi logikanya: "Lanjutkan (ok = true) selama x masih di bawah target". Begitu x menyentuh atau melewati target, ok jadi false dan program berhenti.

3. **Tugas 3**

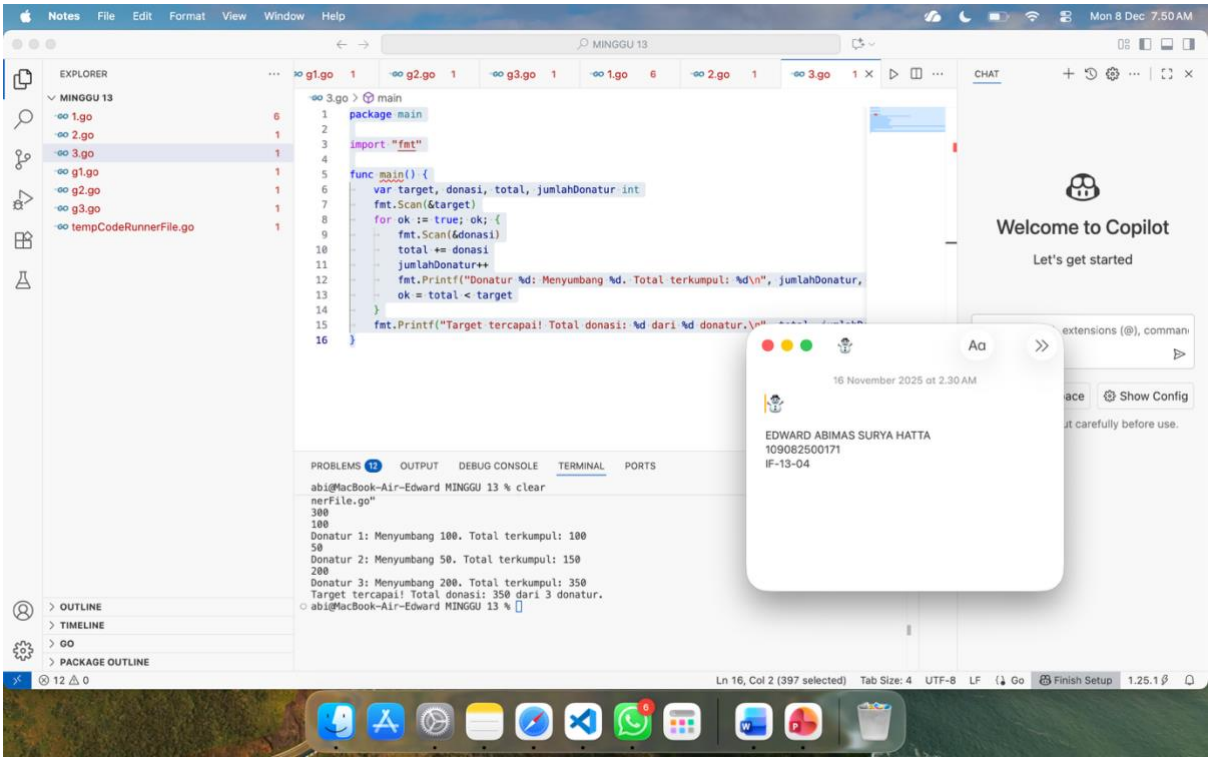
**Source code**

```
package main

import "fmt"

func main() {
    var target, donasi, total, jumlahDonatur int
    fmt.Scan(&target)
    for ok := true; ok; {
        fmt.Scan(&donasi)
        total += donasi
        jumlahDonatur++
        fmt.Printf("Donatur %d: Menyumbang %d. Total
terkumpul: %d\n", jumlahDonatur, donasi, total)
        ok = total < target
    }
    fmt.Printf("Target tercapai! Total donasi: %d dari
%d donatur.\n", total, jumlahDonatur)
}
```

**Screenshoot program**




**Deskripsi program**

Ini adalah simulasi penggalangan dana. Komputer mencatat target donasi yang diinginkan. Lalu masuk ke perulangan yang lagi-lagi dipancing dengan ok := true. Di setiap putaran, komputer meminta input donasi baru. Uang donasi baru itu langsung ditambahkan ke tabungan total. Komputer juga mencatat bahwa ada satu donatur baru (jumlahDonatur ditambah 1). Setelah itu, komputer memberi laporan status saat ini: "Donatur ke-sekian menyumbang sekian, total sekarang sekian". Kemudian komputer mengevaluasi apakah perlu lanjut menggalang dana atau tidak dengan logika: ok = total < target. Artinya, status ok (lanjut) hanya benar jika total uang terkumpul masih di bawah target. Jika totalsudah sama atau melebihi target, maka ok menjadi false. Loop pun berhenti, dan komputer dengan bangga mencetak pesan bahwa target tercapai beserta rekap akhirnya.