LAPORAN PRAKTIKUM

Algoritma Pemrograman

MODUL 04

I/O, TIPE DATA & VARIABEL

Disusun oleh:

EDWARD ABIMAS SURYA HATTA

109082500171

S1IF-13-04

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

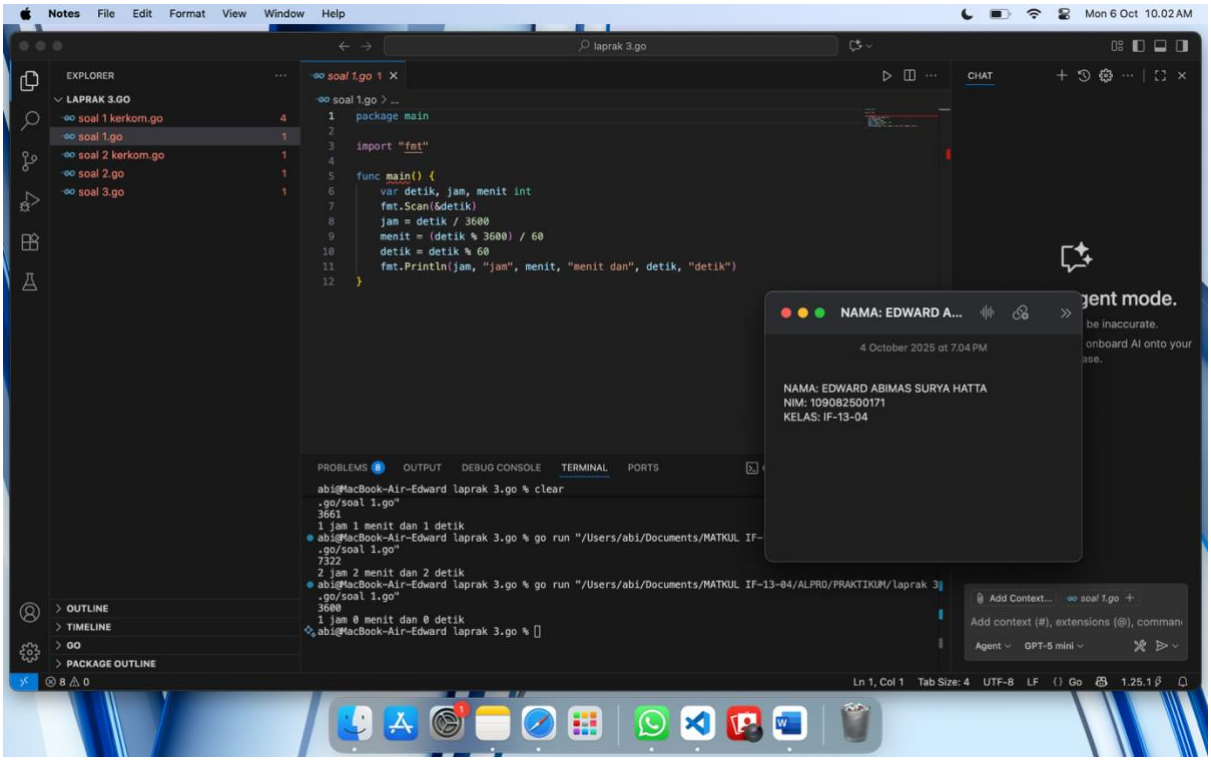
LATIHAN KELAS – GUIDED

1. **Guided 1**

Source Code

```
package main


import "fmt"


func main() {
    var detik, jam, menit int
    fmt.Scan(&detik)
    jam = detik / 3600
    menit = (detik % 3600) / 60
    detik = detik % 60
    fmt.Println(jam, "jam", menit, "menit dan", detik, "detik")
}
```

Screenshoot program



Deskripsi program

Program ini merupakan sebuah aplikasi sederhana yang ditulis dalam bahasa pemrograman Go (Golang) dengan tujuan mengonversi jumlah waktu yang diberikan dalam satuan detik menjadi format waktu yang lebih mudah dipahami oleh manusia, yaitu dalam satuan jam, menit, dan detik. Ketika program dijalankan, ia akan menunggu pengguna memasukkan sebuah angka berupa bilangan bulat yang merepresentasikan total durasi waktu dalam detik. Input tersebut dibaca melalui fungsi `fmt.Scan`, yang menerima alamat memori dari variabel `detik` agar nilai yang dimasukkan pengguna dapat langsung disimpan ke dalam variabel tersebut.

Setelah nilai detik diperoleh, program melakukan serangkaian perhitungan aritmetika untuk memecah total detik tersebut menjadi komponen jam, menit, dan detik yang tersisa. Pertama, jumlah jam dihitung dengan membagi total detik dengan 3600, karena satu jam setara dengan 3600 detik. Karena operasi ini dilakukan antara dua bilangan bulat (`int`), hasilnya secara otomatis berupa bilangan bulat tanpa desimal—artinya bagian pecahan dibuang, sehingga hanya jam penuh yang dihitung. Selanjutnya, untuk menghitung menit, program terlebih dahulu menghitung sisa detik setelah jam diambil menggunakan operator modulo (`%`) terhadap 3600, lalu hasilnya dibagi 60 (karena satu menit terdiri dari 60 detik). Hasil pembagian ini juga berupa bilangan bulat, sehingga hanya menit penuh yang diambil. Terakhir, sisa detik yang tidak cukup membentuk satu menit penuh dihitung dengan mengambil sisa bagi total detik terhadap 60, dan nilai ini kemudian disimpan kembali ke variabel `detik`, menggantikan nilai awal yang telah dimasukkan pengguna.

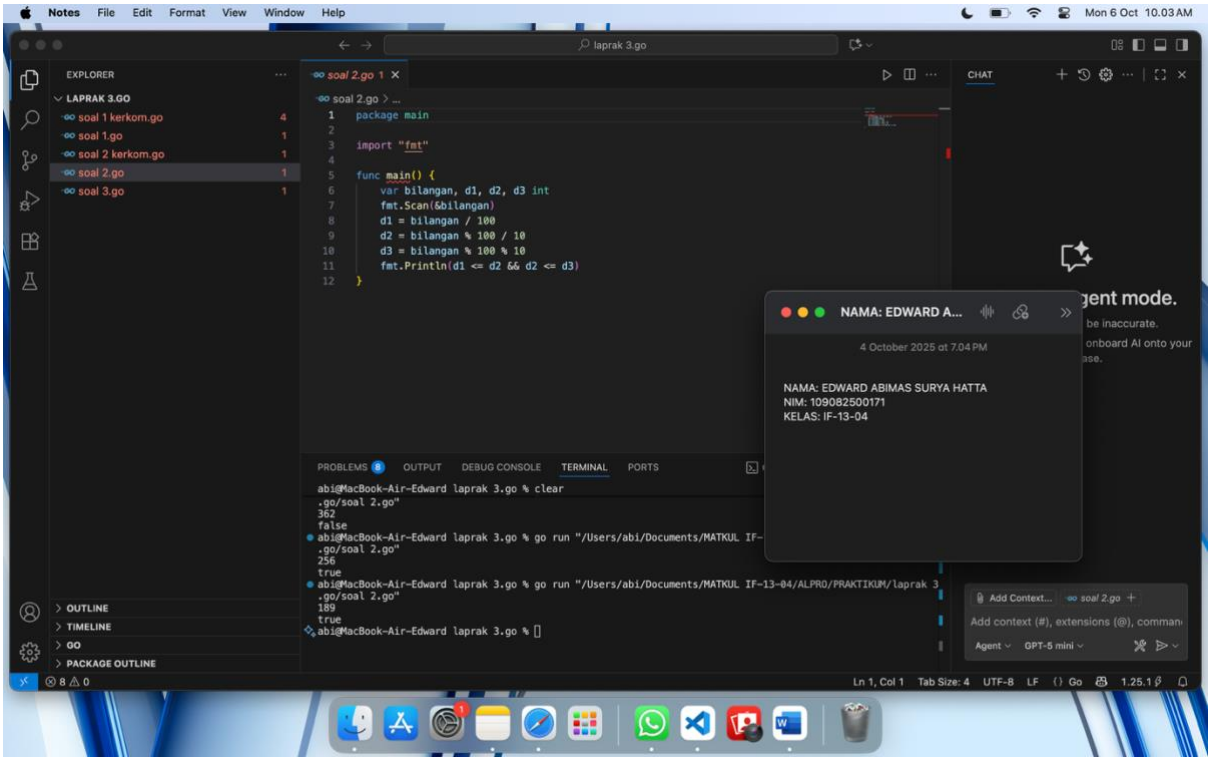
Setelah ketiga komponen waktu—jam, menit, dan detik—diperoleh, program menampilkannya ke layar dalam format kalimat yang mudah dibaca. Fungsi `fmt.Println` digunakan untuk mencetak nilai-nilai tersebut bersama dengan teks penjelas seperti "jam", "menit dan", dan "detik". Go secara otomatis menyisipkan spasi di antara setiap argumen yang diberikan ke `Println`, sehingga output yang dihasilkan terlihat rapi dan alami. Sebagai contoh, jika pengguna memasukkan angka 3661, program akan menghitung bahwa 3661 detik setara dengan 1 jam, 1 menit, dan 1 detik, lalu mencetak kalimat: `1 jam 1 menit dan 1 detik`. Keseluruhan logika program dirancang secara efisien, menggunakan hanya operasi dasar matematika dan tanpa struktur kontrol yang rumit, sehingga mudah dipahami dan dijalankan untuk kebutuhan konversi waktu sederhana.

2. **Guided 2**

   Source Code

```
package main


import "fmt"


func main() {
    var bilangan, d1, d2, d3 int
    fmt.Scan(&bilangan)
    d1 = bilangan / 100
    d2 = bilangan % 100 / 10
    d3 = bilangan % 100 % 10
    fmt.Println(d1 <= d2 && d2 <= d3)
}
```

Screenshoot program



Deskripsi program

Program ini merupakan sebuah aplikasi sederhana dalam bahasa Go yang bertujuan untuk memeriksa apakah digit-digit dari sebuah bilangan bulat tiga digit tersusun secara **tidak menurun (non-decreasing order), artinya setiap digit dari kiri ke kanan nilainya sama atau semakin besar. Program dimulai dengan mendeklarasikan empat variabel bertipe bilangan bulat (`int`): `bilangan` untuk menyimpan input**

pengguna, serta `d1`, `d2`, dan `d3` yang masing-masing akan menyimpan digit ratusan, puluhan, dan satuan dari bilangan tersebut.

Ketika program dijalankan, ia menunggu pengguna memasukkan sebuah bilangan melalui input standar. Input ini dibaca menggunakan fungsi `fmt.Scan`, yang menerima alamat memori dari variabel `bilangan` agar nilai yang diketik pengguna langsung disimpan ke dalam variabel tersebut. Diasumsikan bahwa input yang diberikan adalah bilangan bulat tiga digit (misalnya antara 100 hingga 999), meskipun program secara teknis dapat menerima angka di luar rentang tersebut, namun logika ekstraksi digit tetap berlaku berdasarkan posisi desimal.

Setelah bilangan diterima, program mengekstraksi masing-masing digitnya melalui operasi aritmetika. Digit pertama (`d1`), yaitu digit ratusan, diperoleh dengan membagi `bilangan` dengan 100 menggunakan pembagian bilangan bulat. Karena pembagian antar bilangan bulat di Go menghasilkan hasil tanpa desimal (dipotong ke bawah), maka misalnya jika `bilangan` adalah 357, maka `357 / 100` menghasilkan `3`. Selanjutnya, digit kedua (`d2`), yaitu digit puluhan, dihitung dengan terlebih dahulu mengambil sisa bagi `bilangan` terhadap 100 menggunakan operator modulo (`%`), yang menghilangkan digit ratusan, lalu hasilnya dibagi 10. Sebagai contoh, `357 % 100` menghasilkan `57`, dan `57 / 10` menghasilkan `5`. Terakhir, digit ketiga (`d3`), yaitu digit satuan, diperoleh dengan mengambil `bilangan % 100` terlebih dahulu (yang memberi dua digit terakhir), lalu mengambil sisa bagi hasil tersebut terhadap 10. Dalam contoh yang sama, `357 % 100` adalah `57`, lalu `57 % 10` menghasilkan `7`. Alternatif yang lebih langsung sebenarnya adalah `bilangan % 10`, tetapi cara yang digunakan dalam kode ini tetap menghasilkan nilai yang benar.

Setelah ketiga digit berhasil diekstraksi dan disimpan dalam `d1`, `d2`, dan `d3`, program kemudian mengevaluasi ekspresi logika `d1 <= d2 && d2 <= d3`. Ekspresi ini memeriksa apakah digit pertama kurang dari atau sama dengan digit kedua, **dan digit kedua kurang dari atau sama dengan digit ketiga. Jika kedua kondisi tersebut terpenuhi, maka seluruh ekspresi bernilai `true`; jika salah satu saja tidak terpenuhi, hasilnya adalah `false`. Hasil dari ekspresi boolean ini—entah `true` atau `false`—langsung dicetak ke layar menggunakan `fmt.Println`. Sebagai ilustrasi, jika pengguna memasukkan `123`, maka `d1=1`, `d2=2`, `d3=3`, sehingga `1 <= 2 && 2 <= 3` bernilai `true`, dan program mencetak `true`. Namun jika inputnya `132`, maka `d1=1`, `d2=3`, `d3=2`, sehingga `3 <= 2` bernilai `false`, dan keseluruhan ekspresi menghasilkan `false`.**

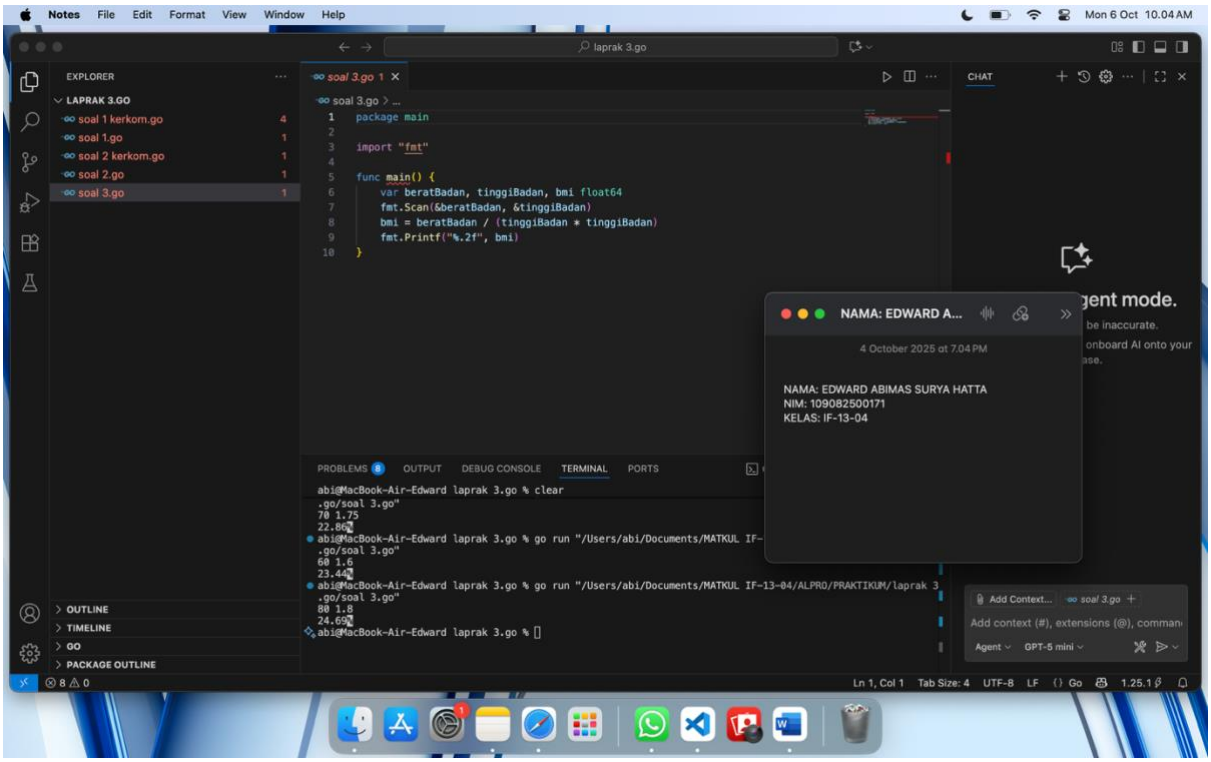
Dengan demikian, program ini berfungsi sebagai pemeriksa urutan digit pada bilangan tiga digit, memberikan jawaban boolean yang menunjukkan apakah digit-digit tersebut tersusun dari kecil ke besar (atau setidaknya tidak menurun) dari kiri ke kanan.

3. **Guided 3**

Source Code

```
package main


import "fmt"


func main() {
    var beratBadan, tinggiBadan, bmi float64
    fmt.Scan(&beratBadan, &tinggiBadan)
    bmi = beratBadan / (tinggiBadan * tinggiBadan)
    fmt.Printf("%.2f", bmi)
}
```

Screenshoot program



Deskripsi program

Program ini merupakan sebuah aplikasi sederhana yang ditulis dalam bahasa pemrograman Go untuk menghitung nilai Indeks Massa Tubuh (Body Mass Index atau BMI) berdasarkan input berat badan dan tinggi badan yang diberikan oleh pengguna. BMI merupakan ukuran standar yang digunakan untuk menilai apakah

berat badan seseorang berada dalam kategori sehat relatif terhadap tinggi badannya. Program dimulai dengan mendeklarasikan tiga variabel bertipe `float64`, yaitu `beratBadan`, `tinggiBadan`, dan `bmi`. Penggunaan tipe data `float64` dipilih karena perhitungan BMI melibatkan angka desimal, dan tipe ini menyediakan presisi yang cukup tinggi untuk representasi bilangan pecahan dalam komputasi ilmiah maupun sehari-hari.

Ketika program dijalankan, ia menunggu pengguna memasukkan dua nilai numerik yang dipisahkan oleh spasi, enter, atau karakter whitespace lainnya. Nilai pertama diasumsikan sebagai berat badan (dalam kilogram), dan nilai kedua sebagai tinggi badan (dalam meter). Input ini dibaca secara bersamaan menggunakan fungsi `fmt.Scan`, yang menerima alamat memori dari kedua variabel tersebut melalui operator `&`, sehingga nilai yang dimasukkan langsung disimpan ke dalam `beratBadan` dan `tinggiBadan` sesuai urutannya.

Setelah kedua nilai diterima, program menghitung BMI menggunakan rumus standar: **BMI = berat badan / (tinggi badan × tinggi badan). Dalam kode, operasi ini dituliskan sebagai `beratBadan / (tinggiBadan * tinggiBadan)`, di mana perkalian `tinggiBadan * tinggiBadan` dilakukan terlebih dahulu karena berada dalam tanda kurung, sesuai aturan prioritas operator matematika. Hasil perhitungan ini kemudian disimpan ke dalam variabel `bmi`.**

Terakhir, program menampilkan nilai BMI yang telah dihitung ke layar dengan format yang terkontrol. Alih-alih menggunakan `fmt.Println` yang mencetak nilai apa adanya, program ini menggunakan `fmt.Printf` dengan spesifikasi format `"%.2f"`. Spesifikasi ini berarti bahwa nilai `bmi` akan dicetak sebagai bilangan pecahan dengan tepat **dua angka di belakang koma, tanpa pembulatan berlebih atau tampilan eksponensial. Misalnya, jika hasil perhitungan BMI adalah `22.666666...`, maka output yang ditampilkan adalah `22.67`. Format ini membuat hasilnya lebih rapi, mudah dibaca, dan sesuai dengan konvensi umum pelaporan BMI dalam konteks kesehatan.**

Secara keseluruhan, program ini menjalankan alur yang sangat lurus: menerima input, melakukan perhitungan sederhana berdasarkan rumus ilmiah yang baku, lalu menampilkan hasilnya dalam format yang konsisten dan informatif. Meskipun tidak menyertakan validasi input (seperti memastikan tinggi badan tidak nol atau nilai-nilai tidak negatif), program ini berfungsi dengan baik selama pengguna memasukkan data yang valid sesuai ekspektasi, yaitu berat badan dalam kilogram dan tinggi badan dalam meter sebagai bilangan positif.

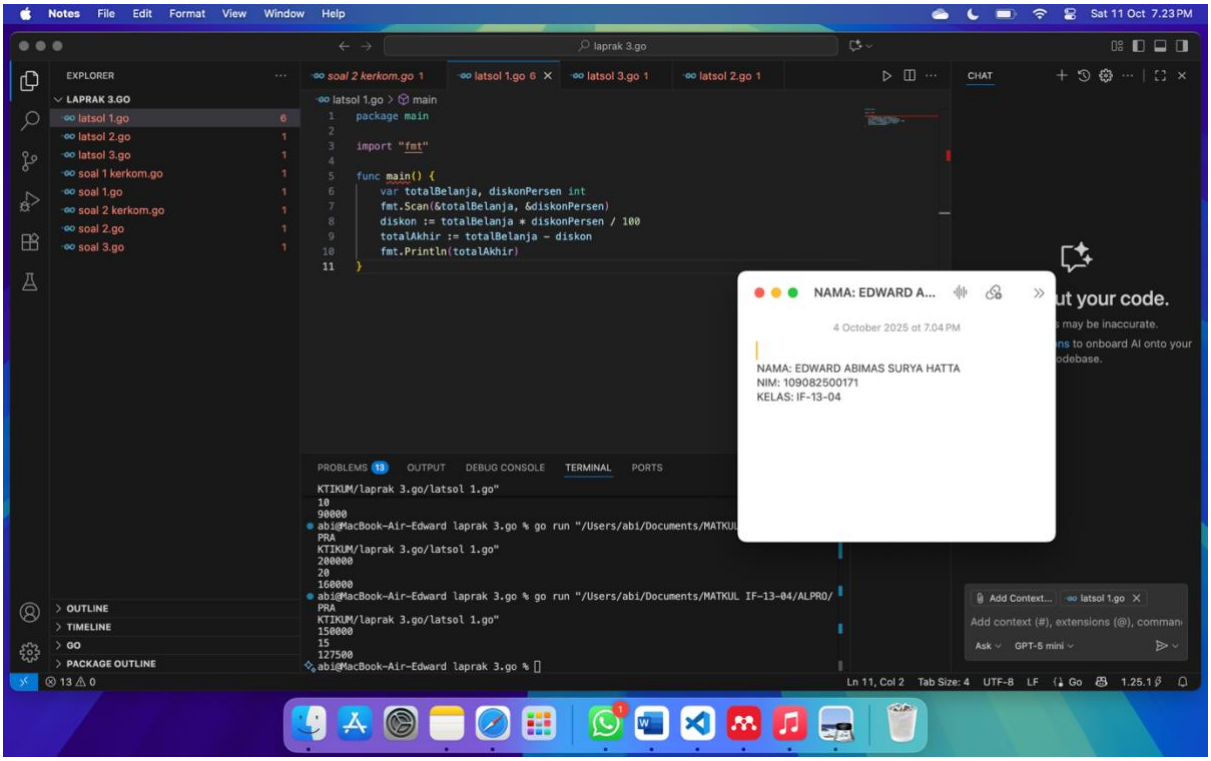
TUGAS

1. **Tugas 1**

   Source code

```go
package main


import "fmt"


func main() {
    var totalBelanja, diskonPersen int
    fmt.Scan(&totalBelanja, &diskonPersen)
    diskon := totalBelanja * diskonPersen / 100
    totalAkhir := totalBelanja - diskon
    fmt.Println(totalAkhir)
}
```

Screenshoot program



Deskripsi program

Program ini merupakan sebuah aplikasi sederhana yang ditulis dalam bahasa Go untuk menghitung total pembayaran akhir setelah diberikan potongan diskon dalam persentase terhadap total belanja awal. Program dimulai dengan mendeklarasikan dua variabel bertipe bilangan bulat (`int`), yaitu `totalBelanja` yang menyimpan nilai total harga barang atau jasa sebelum diskon, dan `diskonPersen` yang menyimpan

besar diskon dalam bentuk persentase (misalnya 10 untuk 10%). Kedua variabel ini akan diisi berdasarkan input yang diberikan oleh pengguna saat program dijalankan.

Ketika program dieksekusi, ia menunggu pengguna memasukkan dua angka bulat yang dipisahkan oleh spasi, tab, atau enter. Input pertama diasumsikan sebagai total belanja dalam satuan mata uang (misalnya rupiah), dan input kedua sebagai persentase diskon yang berlaku. Fungsi `fmt.Scan` digunakan untuk membaca kedua nilai tersebut secara berurutan, dengan masing-masing nilai disimpan langsung ke dalam alamat memori variabel `totalBelanja` dan `diskonPersen` melalui operator `&`, sehingga nilai-nilai tersebut tersedia untuk perhitungan selanjutnya.

Setelah nilai input diterima, program menghitung nominal diskon dalam satuan mata uang. Perhitungan ini dilakukan dengan mengalikan `totalBelanja` dengan `diskonPersen`, lalu membagi hasilnya dengan 100. Karena semua variabel yang terlibat bertipe `int`, operasi aritmetika ini dilakukan dalam domain bilangan bulat, artinya hasil perkalian dan pembagian tidak menghasilkan angka desimal—jika ada sisa pembagian, bagian pecahan tersebut secara otomatis dibuang (bukan dibulatkan). Misalnya, jika total belanja adalah 1500 dan diskon 10%, maka `1500 * 10 / 100` menghasilkan `150`, yang merupakan nilai diskon dalam satuan mata uang.

Selanjutnya, program menghitung total pembayaran akhir dengan mengurangkan nilai diskon dari total belanja awal, dan menyimpan hasilnya ke dalam variabel baru bernama `totalAkhir`. Variabel ini juga bertipe `int`, sehingga hasil akhir tetap berupa bilangan bulat. Terakhir, program mencetak nilai `totalAkhir` ke layar menggunakan fungsi `fmt.Println`, yang secara otomatis menambahkan baris baru setelah output ditampilkan.

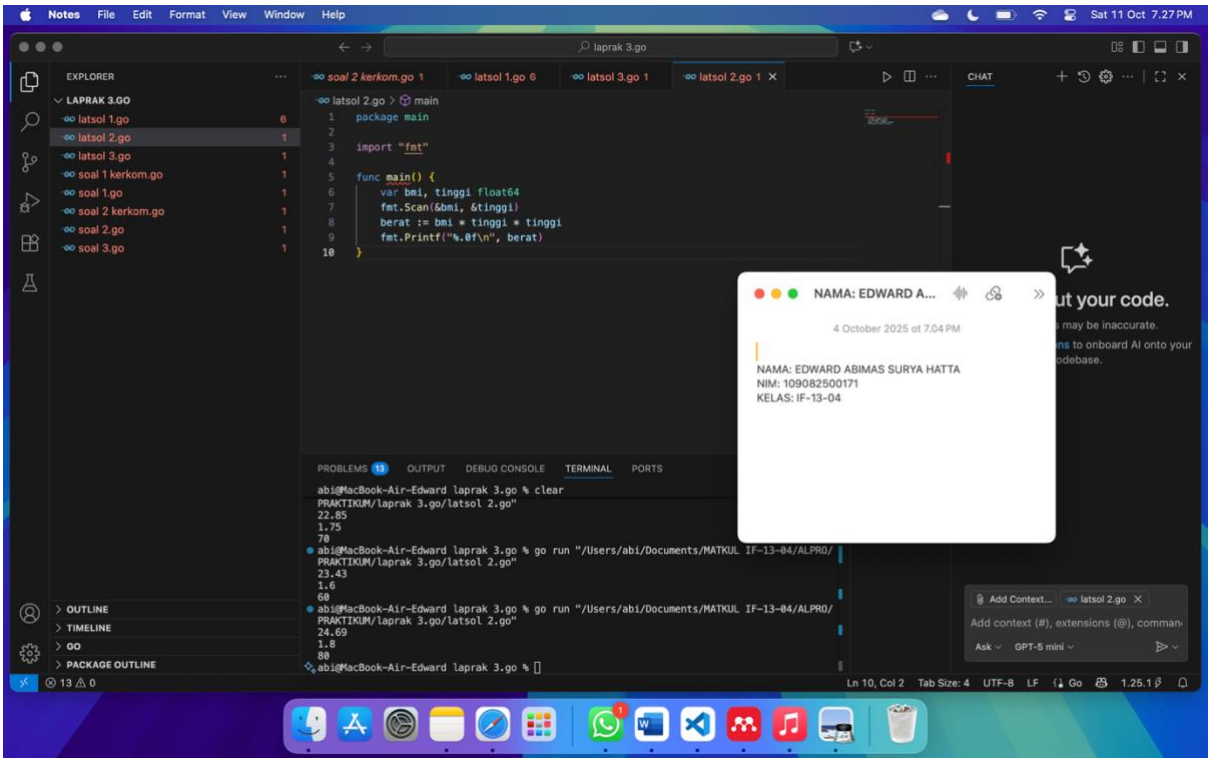
Sebagai contoh, jika pengguna memasukkan `20000 15`, maka program akan menghitung diskon sebesar `20000 * 15 / 100 = 3000`, lalu total akhir menjadi `20000 - 3000 = 17000`, dan angka `17000` itulah yang dicetak sebagai output. Program ini dirancang secara efisien dan langsung, tanpa validasi tambahan atau format output khusus, sehingga sangat cocok untuk skenario sederhana di mana input selalu valid dan hasil diharapkan dalam bentuk bilangan bulat utuh, seperti dalam sistem kasir atau simulasi perhitungan diskon dasar.

2. Tugas 2

Source code

```
package main


import "fmt"


func main() {
    var bmi, tinggi float64
    fmt.Scan(&bmi, &tinggi)
    berat := bmi * tinggi * tinggi
    fmt.Printf("%.0f\n", berat)
}
```

Screenshoot program



Deskripsi program

Program ini merupakan sebuah aplikasi sederhana dalam bahasa Go yang bertujuan untuk menghitung berat badan seseorang berdasarkan nilai Indeks Massa Tubuh (BMI) dan tinggi badan yang diberikan sebagai input. Berbeda dengan perhitungan BMI biasa yang menggunakan berat dan tinggi untuk menghasilkan indeks, program ini melakukan proses sebaliknya: dari nilai BMI dan tinggi badan, ia menghitung berapa berat badan yang sesuai dengan kombinasi kedua nilai tersebut.

Program dimulai dengan mendeklarasikan dua variabel bertipe `float64`, yaitu `bmi` dan `tinggi`. Tipe data `float64` dipilih karena baik BMI maupun tinggi badan umumnya melibatkan angka pecahan (desimal), dan tipe ini memberikan presisi yang cukup untuk perhitungan ilmiah maupun medis dasar. Ketika program dijalankan, ia menunggu pengguna memasukkan dua nilai numerik yang dipisahkan oleh spasi, tab, atau enter. Nilai pertama diasumsikan sebagai nilai BMI (misalnya 22.5), dan nilai kedua sebagai tinggi badan dalam satuan meter (misalnya 1.75). Input ini dibaca secara bersamaan menggunakan fungsi `fmt.Scan`, yang menerima alamat memori dari kedua variabel melalui operator `&`, sehingga nilai-nilai tersebut langsung disimpan ke dalam `bmi` dan `tinggi` sesuai urutannya.

Setelah kedua nilai diterima, program menghitung berat badan menggunakan rumus yang merupakan inversi dari rumus BMI standar. Rumus BMI adalah `BMI = berat / (tinggi²)`, sehingga jika dirumuskan ulang untuk mencari berat, menjadi `berat = BMI × tinggi²`. Dalam kode, hal ini diwujudkan melalui ekspresi `bmi * tinggi * tinggi`, yang secara matematis setara dengan `bmi × tinggi²`. Hasil perhitungan ini disimpan dalam variabel baru bernama `berat`, yang secara implisit juga bertipe `float64` karena semua operand dalam ekspresi tersebut adalah bilangan pecahan.

Kemudian, program menampilkan hasil perhitungan berat badan ke layar menggunakan fungsi `fmt.Printf` dengan spesifikasi format `"%.0f\n"`. Spesifikasi ini memiliki dua bagian penting: `%.0f` berarti bahwa nilai `berat` akan diformat sebagai bilangan pecahan (`f` untuk floating-point) dengan **nol digit di belakang koma, sehingga hasilnya dibulatkan ke bilangan bulat terdekat tanpa menampilkan titik desimal. Karakter `\n` di akhir string format menjamin bahwa setelah mencetak hasil, kursor berpindah ke baris berikutnya, sesuai dengan praktik umum dalam output terminal. Sebagai contoh, jika perhitungan menghasilkan `68.73`, maka output yang ditampilkan adalah `69`; jika hasilnya `68.49`, maka yang ditampilkan adalah `68`.**

Secara keseluruhan, program ini menjalankan logika terbalik dari kalkulator BMI konvensional, memberikan estimasi berat badan ideal atau aktual berdasarkan indeks dan tinggi yang diketahui. Meskipun tidak menyertakan validasi input—seperti memastikan bahwa BMI dan tinggi bernilai positif—program ini berfungsi dengan akurat selama pengguna memasukkan data yang masuk akal, yaitu BMI dalam rentang medis yang wajar (misalnya 15–40) dan tinggi badan dalam meter (misalnya 1.50 hingga 2.00). Outputnya disajikan dalam bentuk bilangan bulat untuk

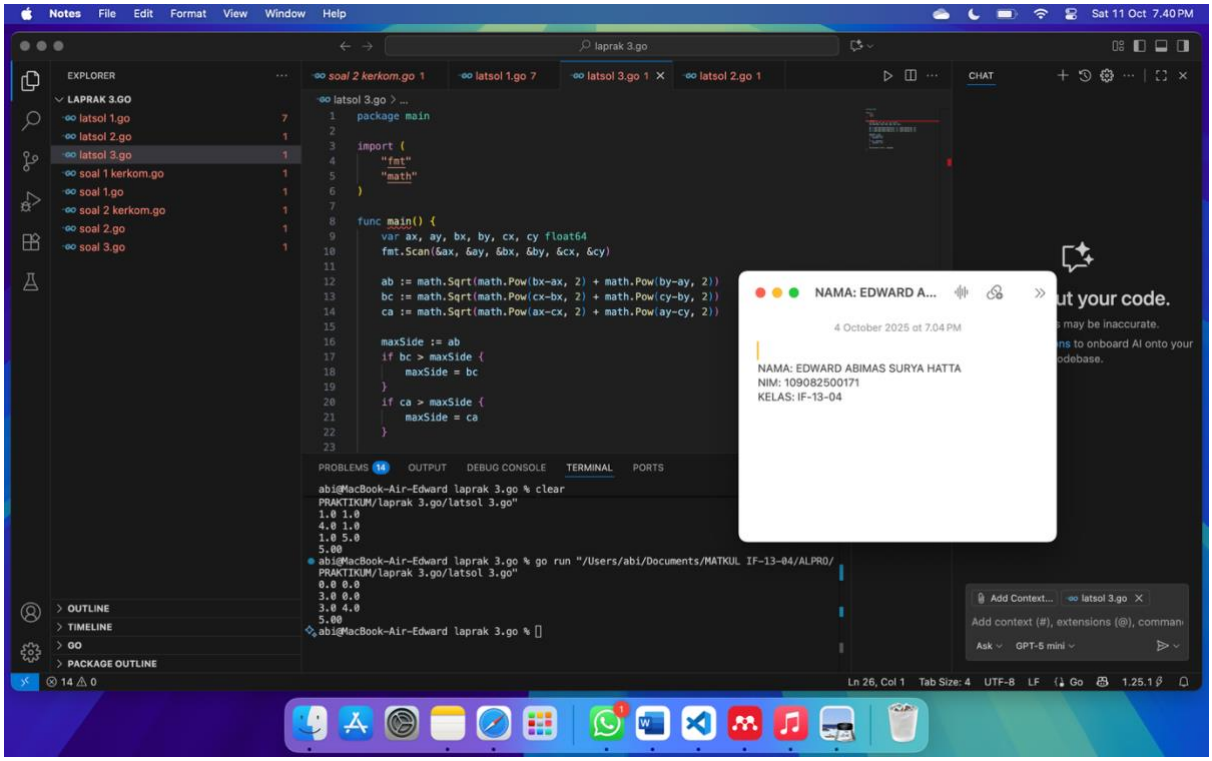
menyederhanakan interpretasi, karena berat badan dalam konteks sehari-hari umumnya dilaporkan tanpa desimal.

3. **Tugas 3**

Source code

```
package main

import (
    "fmt"
    "math"
)

func main() {
    var ax, ay, bx, by, cx, cy float64
    fmt.Scan(&ax, &ay, &bx, &by, &cx, &cy)

    ab := math.Sqrt(math.Pow(bx-ax, 2) + math.Pow(by-
ay, 2))
    bc := math.Sqrt(math.Pow(cx-bx, 2) + math.Pow(cy-
by, 2))
    ca := math.Sqrt(math.Pow(ax-cx, 2) + math.Pow(ay-
cy, 2))

    maxSide := ab
    if bc > maxSide {
        maxSide = bc
    }
    if ca > maxSide {
        maxSide = ca
    }

    fmt.Printf("%.2f\n", maxSide)
}
```

Screenshoot program



Deskripsi program

Program ini merupakan sebuah aplikasi dalam bahasa Go yang dirancang untuk menghitung panjang sisi terpanjang dari sebuah segitiga yang dibentuk oleh tiga titik pada bidang koordinat dua dimensi. Ketiga titik tersebut diberikan sebagai pasangan koordinat: titik A dengan koordinat (ax, ay), titik B dengan koordinat (bx, by), dan titik C dengan koordinat (cx, cy). Program membaca keenam nilai koordinat ini dari input pengguna, lalu menghitung jarak antara setiap pasangan titik untuk mendapatkan panjang ketiga sisi segitiga, yaitu sisi AB, BC, dan CA. Setelah itu, program menentukan sisi mana yang paling panjang di antara ketiganya dan mencetak nilainya dengan format dua angka di belakang koma.

Program dimulai dengan mengimpor dua package standar Go: `fmt` untuk menangani operasi masukan dan keluaran, serta `math` yang menyediakan fungsi-fungsi matematika seperti akar kuadrat (`Sqrt`) dan perpangkatan (`Pow`). Di dalam fungsi `main`, dideklarasikan enam variabel bertipe `float64`—`ax`, `ay`, `bx`, `by`, `cx`, dan `cy`—untuk menyimpan koordinat x dan y dari masing-masing titik. Tipe data `float64` digunakan karena koordinat titik dalam bidang kartesius sering kali berupa bilangan pecahan, dan perhitungan jarak memerlukan presisi tinggi.

Ketika program dijalankan, ia menunggu pengguna memasukkan enam bilangan real yang masing-masing merepresentasikan koordinat titik A, B, dan C secara berurutan. Input ini dibaca sekaligus menggunakan `fmt.Scan`, yang mengisi nilai-nilai tersebut langsung ke variabel-variabel yang sesuai berdasarkan urutan alamat memorinya.

Setelah koordinat diperoleh, program menghitung panjang ketiga sisi segitiga menggunakan rumus jarak Euclidean antara dua titik di bidang datar. Rumus jarak antara dua titik \((x_1, y_1)\) dan \((x_2, y_2)\) adalah \(\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}\). Untuk sisi AB, program menghitung selisih koordinat x dan y antara titik B dan A, menguadratkannya masing-masing menggunakan `math.Pow`, menjumlahkannya, lalu mengambil akar kuadrat dari hasil tersebut dengan `math.Sqrt`. Proses serupa dilakukan untuk sisi BC (antara B dan C) dan sisi CA (antara C dan A), dengan hasil masing-masing disimpan dalam variabel `ab`, `bc`, dan `ca`.

Selanjutnya, program menentukan sisi terpanjang di antara ketiganya. Inisialisasi dimulai dengan mengasumsikan bahwa sisi AB (`ab`) adalah yang terpanjang, lalu membandingkannya secara berurutan dengan sisi BC dan CA. Jika ditemukan sisi lain yang lebih panjang, nilai variabel `maxSide` diperbarui untuk menyimpan panjang sisi tersebut. Pendekatan ini menggunakan struktur kondisional sederhana tanpa fungsi `math.Max` berulang, sehingga mudah dipahami dan efisien.

Terakhir, program mencetak nilai `maxSide`—yaitu panjang sisi terpanjang segitiga—dengan format dua angka di belakang koma menggunakan `fmt.Printf` dan spesifikasi format `"%.2f\n"`. Karakter `\n` memastikan output diakhiri dengan baris baru, sesuai standar keluaran teks. Sebagai contoh, jika tiga titik membentuk segitiga dengan sisi 3.0, 4.0, dan 5.0, maka program akan mencetak `5.00`. Dengan demikian, program ini memberikan solusi akurat dan ringkas untuk menentukan sisi terpanjang segitiga berdasarkan koordinat titik-titik sudutnya.

TUGAS PENDAHULUAN

SOAL 1

PEMBERI SOAL: MICHAEL YEREMIA SUCIYONO – 109082500180

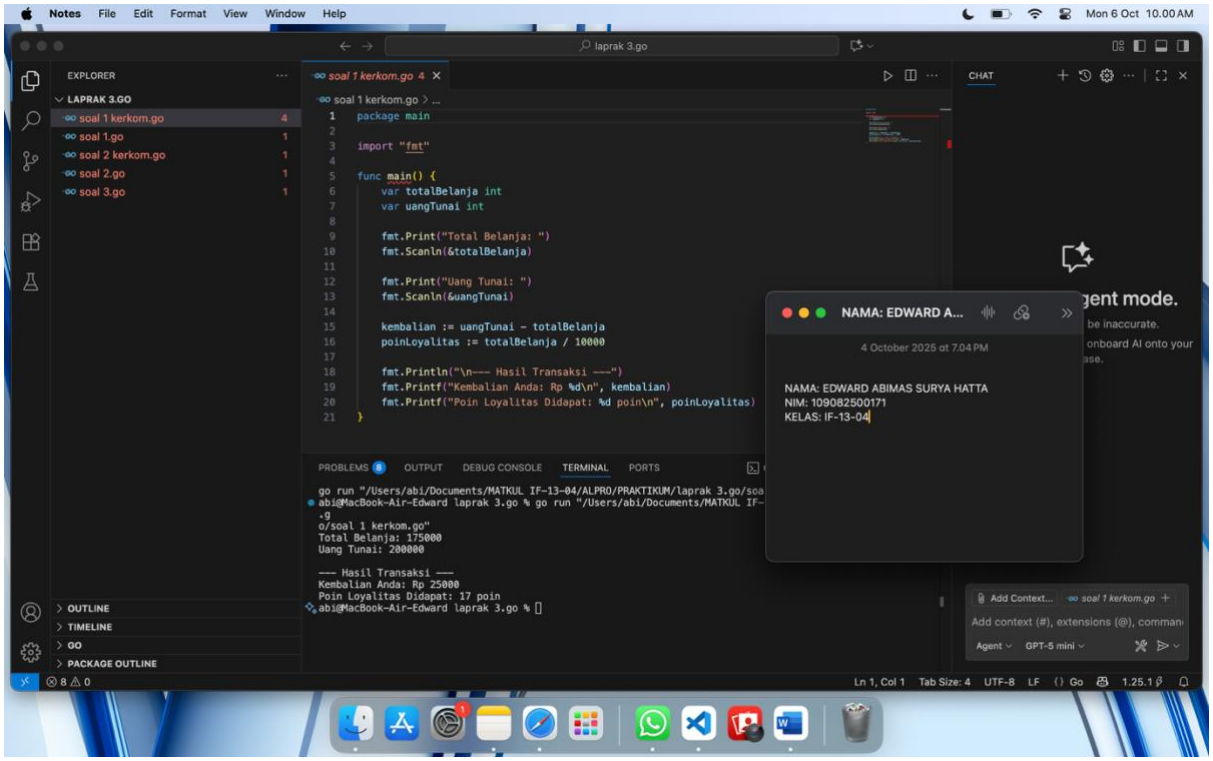
DESKRIPSI SOAL: Buatlah program kasir sederhana. Program ini harus menghitung total kembalian uang dan menentukan poin loyalitas yang didapatkan pelanggan. Poin didapatkan hanya dari kelipatan penuh Rp 10.000 dari total belanja

INPUT	OUTPUT
TOTAL BELANJA: 175000 **UANG TUNAI: 200000**	**--- Hasil Transaksi ---** **Kembalian Anda: Rp 25000** **Poin Loyalitas Didapat: 17 poin**

SOURCE CODE:

```
package main

import "fmt"

func main() {
    var totalBelanja int
    var uangTunai int

    fmt.Print("Total Belanja: ")
    fmt.Scanln(&totalBelanja)

    fmt.Print("Uang Tunai: ")
    fmt.Scanln(&uangTunai)

    kembalian := uangTunai - totalBelanja
    poinLoyalitas := totalBelanja / 10000

    fmt.Println("\n--- Hasil Transaksi ---")
    fmt.Printf("Kembalian Anda: Rp %d\n", kembalian)
    fmt.Printf("Poin Loyalitas Didapat: %d poin\n",
poinLoyalitas)
}
```

SCREENSHOT HASIL EKSEKUSI:



SOAL 2

PEMBERI SOAL: MICHAEL YEREMIA SUCIYONO – 109082500180

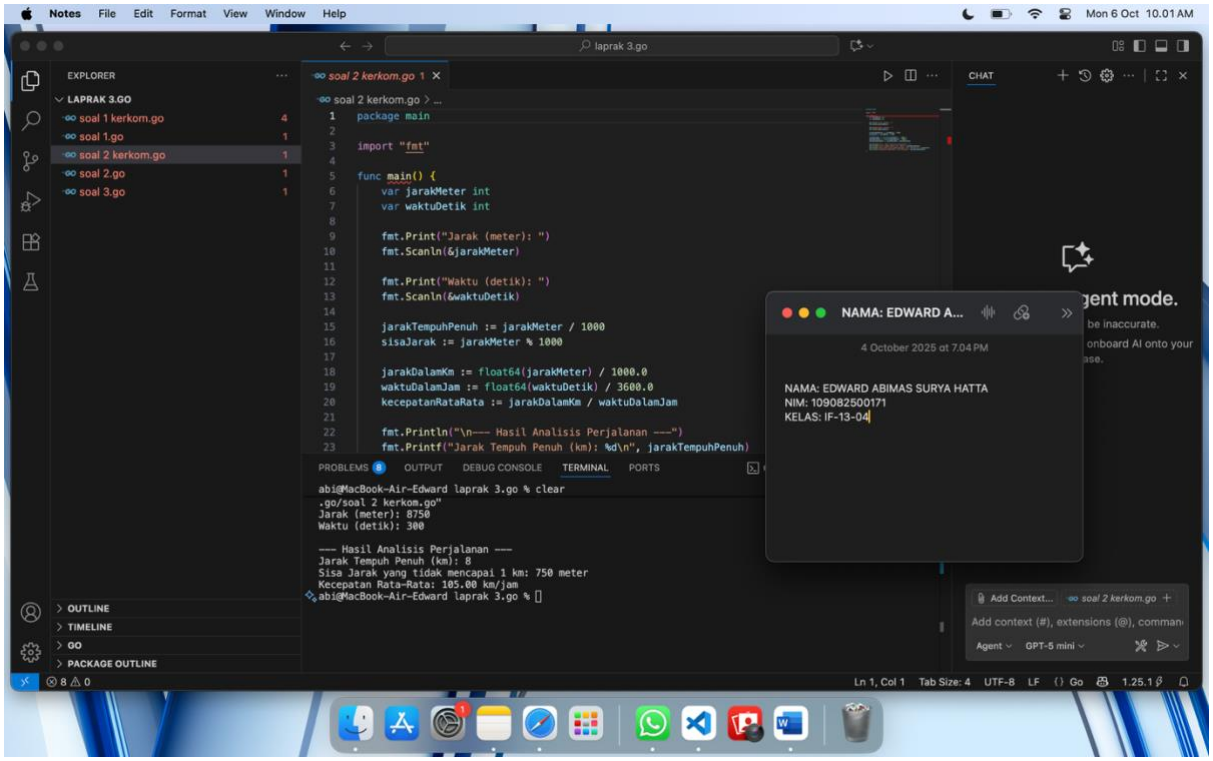
DESKRIPSI SOAL: Anda diminta membuat program yang menghitung kecepatan rata-rata kendaraan dalam satuan kilometer per jam (km/jam) dari input jarak dalam meter dan waktu dalam detik. Program juga harus menghitung sisa jarak tempuh yang tidak mencapai kelipatan penuh 1 kilometer.

INPUT	OUTPUT
JARAK (METER): 8750 **WAKTU (DETIK): 300**	**--- Hasil Analisis Perjalanan ---** **Jarak Tempuh Penuh (km): 8** **Sisa Jarak yang tidak mencapai 1 km: 750 meter** **Kecepatan Rata-Rata: 105.00 km/jam**

SOURCE CODE:

```
package main

import "fmt"

func main() {
    var jarakMeter int
    var waktuDetik int

    fmt.Print("Jarak (meter): ")
    fmt.Scanln(&jarakMeter)

    fmt.Print("Waktu (detik): ")
    fmt.Scanln(&waktuDetik)

    jarakTempuhPenuh := jarakMeter / 1000
    sisaJarak := jarakMeter % 1000

    jarakDalamKm := float64(jarakMeter) / 1000.0
    waktuDalamJam := float64(waktuDetik) / 3600.0
    kecepatanRataRata := jarakDalamKm / waktuDalamJam

    fmt.Println("\n--- Hasil Analisis Perjalanan ---")
    fmt.Printf("Jarak Tempuh Penuh (km): %d\n",
jarakTempuhPenuh)
    fmt.Printf("Sisa Jarak yang tidak mencapai 1 km: %d
meter\n", sisaJarak)
    fmt.Printf("Kecepatan Rata-Rata: %.2f km/jam\n",
kecepatanRataRata)
}
```

SCREENSHOT HASIL EKSEKUSI:


Notes File Edit Format View Window Help
Mon 6 Oct 10.01 AM
laprak 3.go
EXPLORER
LAPRAK 3.GO
soal 1 kerkom.go 4
soal 1.go 1
soal 2 kerkom.go 1
soal 2.go 1
soal 3.go 1
soal 2 kerkom.go 1
soal 2 kerkom.go > ...
1 package main
3 import "fmt"
5 func main() {
6 var jarakMeter int
7 var waktuDetik int
9 fmt.Print("Jarak (meter): ")
10 fmt.Scanln(&jarakMeter)
12 fmt.Print("Waktu (detik): ")
13 fmt.Scanln(&waktuDetik)
15 jarakTempuhPenuh := jarakMeter / 1000
16 sisaJarak := jarakMeter % 1000
18 jarakDalamKm := float64(jarakMeter) / 1000.0
19 waktuDalamJam := float64(waktuDetik) / 3600.0
20 kecepatanRataRata := jarakDalamKm / waktuDalamJam
22 fmt.Println("\n--- Hasil Analisis Perjalanan ---")
23 fmt.Printf("Jarak Tempuh Penuh (km): %d\n", jarakTempuhPenuh)
PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL PORTS
abi@MacBook-Air-Edward laprak 3.go % clear
.go/soal 2 kerkom.go"
Jarak (meter): 8750
Waktu (detik): 300
--- Hasil Analisis Perjalanan ---
Jarak Tempuh Penuh (km): 8
Sisa Jarak yang tidak mencapai 1 km: 750 meter
Kecepatan Rata-Rata: 105.00 km/jam
abi@MacBook-Air-Edward laprak 3.go %
CHAT
gent mode.
be inaccurate.
onboard AI onto your
ase.
NAMA: EDWARD A...
4 October 2025 at 7.04 PM
NAMA: EDWARD ABIMAS SURYA HATTA
NIM: 109082500171
KELAS: IF-13-04
Add Context... soal 2 kerkom.go
Add context (#), extensions (@), comman
Agent GPT-5 mini
OUTLINE
TIMELINE
GO
PACKAGE OUTLINE
8 0
Ln 1, Col 1 Tab Size: 4 UTF-8 LF Go 1.25.1