LAPORAN PRAKTIKUM

Algoritma Pemrograman

EVALUASI

Disusun oleh:

EDWARD ABIMAS SURYA HATTA

109082500171

S1IF-13-04

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

SOAL

1. **SOAL 1**

   Source Code

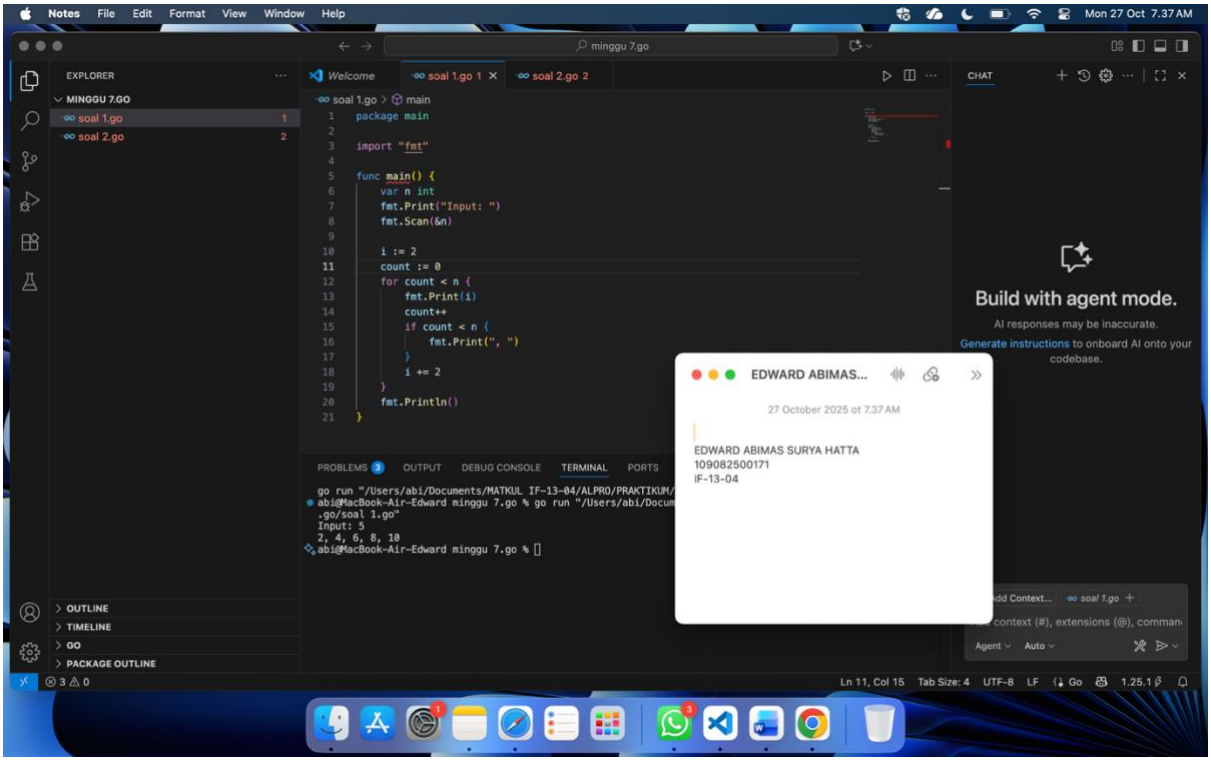
```
package main


import "fmt"


func main() {
    var n int
    fmt.Print("Input: ")
    fmt.Scan(&n)


    i := 2
    count := 0
    for count < n {
        fmt.Print(i)
        count++
        if count < n {
            fmt.Print(", ")
        }
        i += 2
    }
    fmt.Println()
}
```

Screenshoot program



Deskripsi program

Program ini dimulai dengan deklarasi package main, yang mengidentifikasi berkas ini sebagai sebuah program aplikasi yang dapat dieksekusi secara mandiri. Selanjutnya, program mengimpor paket "fmt", yang merupakan pustaka standar Go untuk fungsionalitas *input* dan *output* (I/O) yang diformat.

Titik masuk eksekusi program adalah func main(). Di dalam fungsi ini, baris pertama, var n int, mendeklarasikan sebuah variabel dengan nama n dan tipe data integer. Variabel ini dialokasikan untuk menyimpan jumlah total bilangan genap yang ingin ditampilkan oleh pengguna.

Program kemudian berinteraksi dengan pengguna. fmt.Print("Input: ") dieksekusi untuk menampilkan teks "Input: " pada konsol standar. Fungsi Print digunakan agar kursor tidak berpindah ke baris baru setelah teks ditampilkan. Segera setelah itu, fmt.Scan(&n) dipanggil. Fungsi ini akan menjeda eksekusi program, menunggu pengguna memasukkan data melalui *keyboard* dan menekan tombol Enter. Data yang dimasukkan akan di-parse sebagai integer dan disimpan ke dalam alamat memori dari variabel n (ditunjukkan oleh operator ampersand &).

Setelah nilai n diperoleh, dua variabel diinisialisasi menggunakan deklarasi singkat. i := 2 menginisialisasi variabel idengan nilai awal 2. Variabel i ini akan berfungsi sebagai generator bilangan genap, dimulai dari bilangan genap positif pertama. count := 0 menginisialisasi variabel count sebagai pencacah (counter) untuk melacak berapa banyak bilangan yang telah dicetak.

Inti dari logika program terdapat dalam struktur perulangan for count < n. Perulangan ini akan terus berjalan selama nilai count masih lebih kecil dari nilai n yang dimasukkan pengguna.

Di dalam setiap iterasi perulangan, fmt.Print(i) dieksekusi untuk mencetak nilai i saat ini ke konsol. Setelah pencetakan, count++ dieksekusi, menaikkan nilai pencacah sebesar satu untuk menandai bahwa satu bilangan telah berhasil ditampilkan.

Selanjutnya, terdapat sebuah blok kondisional if count < n. Pengecekan ini sangat penting untuk penataan format keluaran. Jika count masih lebih kecil dari n, ini mengindikasikan bahwa bilangan yang baru saja dicetak bukanlah bilangan terakhir dalam deret. Oleh karena itu, fmt.Print(", ") dieksekusi untuk mencetak koma dan spasi sebagai pemisah sebelum bilangan berikutnya. Jika count sudah sama dengan n, kondisi ini bernilai salah (false), dan koma tidak akan dicetak setelah bilangan terakhir.

Instruksi terakhir di dalam blok perulangan adalah i += 2. Pernyataan ini memperbarui nilai i dengan menambahkannya sebanyak 2, sehingga pada iterasi berikutnya, i akan berisi bilangan genap selanjutnya.

Proses ini (mencetak i, menaikkan count, mengecek koma, menaikkan i) akan berulang hingga count mencapai nilai yang sama dengan n. Pada titik tersebut, kondisi for count < n akan bernilai salah, dan program akan keluar dari perulangan.

Sebagai instruksi penutup, fmt.Println() dieksekusi. Fungsi ini mencetak sebuah karakter baris baru (newline) ke konsol. Tujuannya adalah untuk memindahkan kursor ke baris di bawahnya, memastikan bahwa keluaran program rapi dan *prompt* terminal berikutnya muncul di baris yang terpisah.

2. **SOAL 2**

Source Code

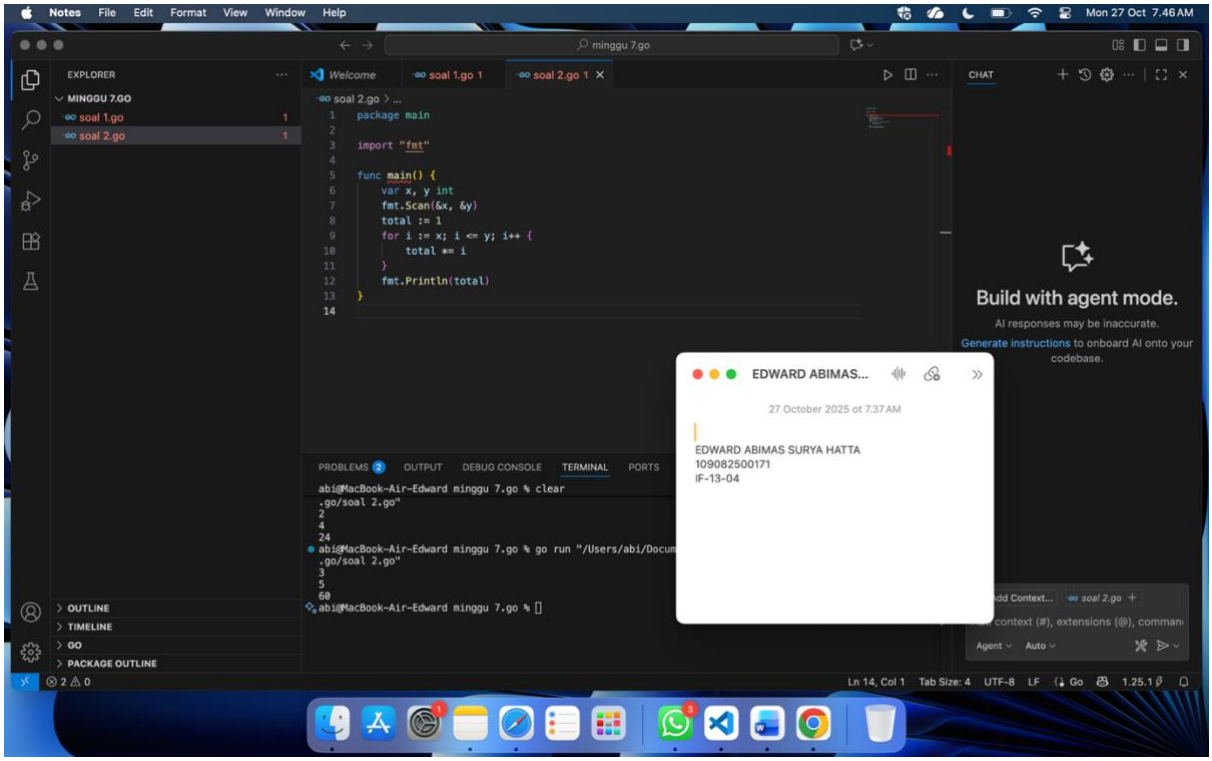
```
package main


import "fmt"


func main() {
    var x, y int
    fmt.Scan(&x, &y)
    total := 1
    for i := x; i <= y; i++ {
        total *= i
    }
    fmt.Println(total)
```

```
}
```

Screenshoot program



Deskripsi program

Program ini dimulai dengan deklarasi package main, yang menandakan bahwa berkas ini adalah sebuah program aplikasi yang dapat dieksekusi. Selanjutnya, program mengimpor paket fmt dari pustaka standar Go. Paket ini sangat penting karena menyediakan fungsi-fungsi yang diperlukan untuk melakukan operasi *input* dan *output* (I/O) yang diformat, seperti membaca masukan dari pengguna (fmt.Scan) dan mencetak keluaran ke konsol (fmt.Println).

Eksekusi program dimulai di dalam func main(). Di sini, var x, y int digunakan untuk mendeklarasikan dua variabel integer (bilangan bulat) yang bernama x dan y. Pada titik ini, keduanya memiliki nilai *default* integer, yaitu 0. Baris berikutnya, fmt.Scan(&x, &y), adalah instruksi untuk program agar berhenti sejenak dan menunggu masukan dari pengguna. Program akan membaca dua nilai integer yang diketikkan pengguna (dipisahkan oleh spasi atau baris baru) dan menyimpannya ke dalam alamat memori dari variabel x dan y, yang ditandai oleh operator ampersand (&).

Setelah mendapatkan nilai x dan y dari pengguna, program menginisialisasi variabel baru menggunakan deklarasi singkat, total := 1. Variabel total ini akan berfungsi sebagai "akumulator" untuk menyimpan hasil perhitungan. Nilai awalnya diatur ke 1, yang merupakan langkah krusial. Dalam matematika, 1 adalah identitas multiplikatif;

ini berarti angka apa pun yang dikalikan dengan 1 akan menghasilkan angka itu sendiri. Jika total dimulai dari 0, hasil akhir dari perkalian beruntun apa pun akan selalu 0.
Inti dari program ini adalah struktur perulangan for i := x; i <= y; i++. Ini adalah *loop* yang akan berjalan melalui setiap angka, dimulai dari x hingga y (inklusif). Bagian i := x adalah inisialisasi, yang menetapkan variabel *counter* i agar dimulai dari nilai x yang dimasukkan pengguna. Bagian i <= y adalah kondisi; *loop* akan terus berlanjut selama nilai i masih lebih kecil atau sama dengan nilai y. Di dalam setiap iterasi *loop*, perintah total *= i dieksekusi. Ini adalah kependekan dari total = total * i. Perintah ini mengambil nilai total saat ini, mengalikannya dengan nilai i saat ini, dan kemudian menyimpan hasilnya kembali ke dalam variabel total. Setelah itu, bagian i++ dieksekusi, yang menaikkan nilai i sebesar 1, sehingga pada iterasi berikutnya, program akan menghitung dengan angka selanjutnya dalam urutan tersebut.
Perulangan ini akan terus berjalan, mengalikan total dengan , kemudian dengan , lalu , dan seterusnya, hingga akhirnya mengalikan total dengan y. Setelah iterasi untuk i = y selesai, i akan dinaikkan menjadi . Pada titik ini, kondisi i <= y akan menjadi salah (false), dan eksekusi *loop* akan berhenti. Akhirnya, program menjalankan fmt.Println(total). Fungsi ini mencetak nilai akhir yang telah terakumulasi di dalam variabel total ke konsol, diikuti dengan sebuah karakter baris baru. Secara efektif, program ini menghitung hasil perkalian dari semua bilangan bulat dari sampai .

3. **SOAL 3**
   Source Code

```
package main


import "fmt"


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


    peti := keping / 800
    sisa := keping % 800


    karung := sisa / 80
    sisa = sisa % 80
```

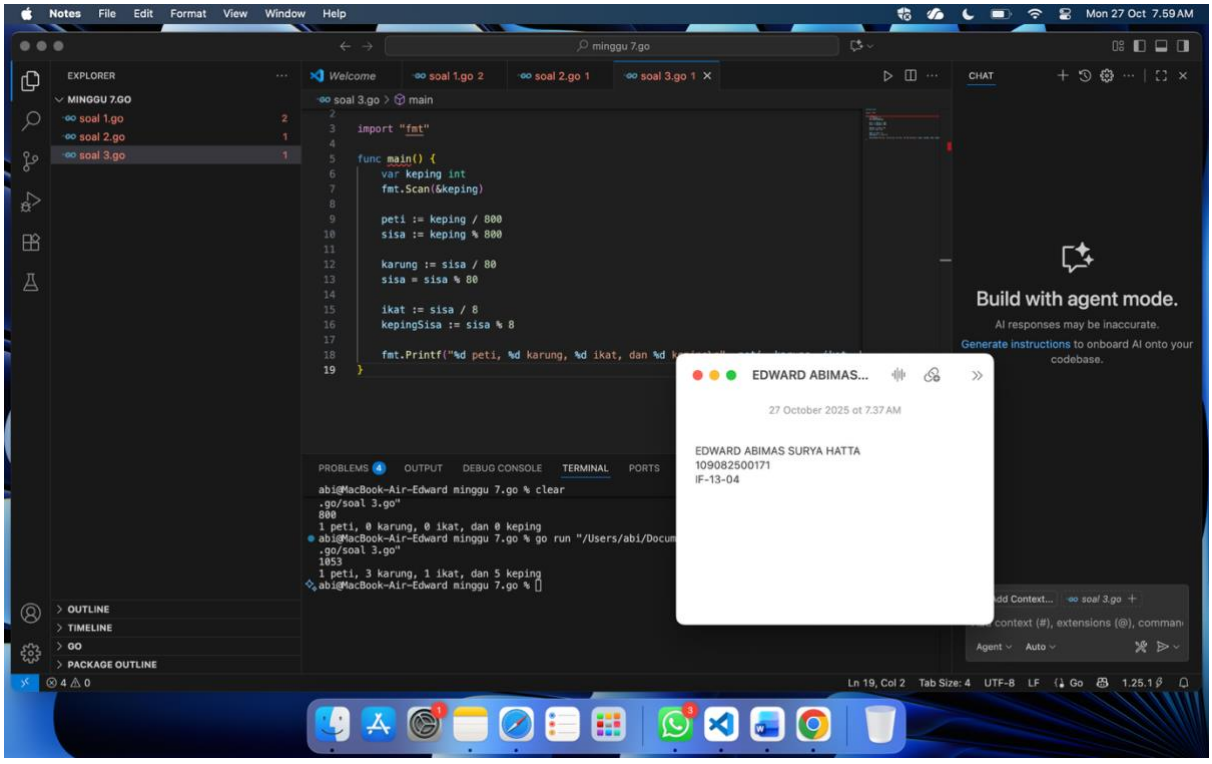
```
        ikat := sisa / 8

        kepingSisa := sisa % 8


        fmt.Printf("%d peti, %d karung, %d ikat, dan %d
    keping\n", peti, karung, ikat, kepingSisa)

    }
```

Screenshoot program



Deskripsi program

Program ini adalah sebuah utilitas konversi yang bertujuan untuk memecah sejumlah total "keping" menjadi beberapa unit pengelompokan yang lebih besar, yaitu "peti", "karung", dan "ikat", serta melaporkan sisa keping yang tidak dapat dikelompokkan. Program dimulai dengan deklarasi package main dan mengimpor paket fmt, yang penting untuk menangani *input* (masukan) dari pengguna dan *output* (keluaran) yang diformat. Di dalam fungsi main, sebuah variabel integer bernama keping dideklarasikan menggunakan var keping int. Program kemudian segera meminta masukan dari pengguna dengan fmt.Scan(&keping), yang akan menjeda eksekusi dan menunggu pengguna memasukkan sebuah angka. Angka tersebut akan disimpan dalam variabel keping.

Logika inti program ini adalah serangkaian operasi pembagian integer (/) dan modulo (%) yang hierarkis, dimulai dari unit terbesar ke unit terkecil. Pertama, program menghitung jumlah "peti" dengan peti := keping / 800. Operasi pembagian integer ini menghitung berapa kali 800 "masuk" secara penuh ke dalam total keping, dan hasilnya disimpan di variabel peti. Segera setelah itu, sisa := keping % 800 menghitung sisa dari pembagian tersebut. Ini adalah jumlah keping yang tersisa setelah semua "peti" penuh telah dipisahkan.

Selanjutnya, program bekerja *hanya* dengan nilai sisa yang baru didapat. karung := sisa / 80 menghitung berapa banyak "karung" (unit 80) yang dapat dibuat dari sisa keping tadi. Setelah jumlah karung didapat, nilai sisa diperbarui dengan sisa = sisa % 80. Perintah ini mengambil sisa sebelumnya (dari pembagian 800) dan mencari sisa *baru* setelah dibagi 80. Dengan demikian, variabel sisa kini menyimpan kepingan yang tidak cukup untuk membentuk "peti" *atau* "karung".

Proses yang sama diulangi untuk unit berikutnya. ikat := sisa / 8 menghitung berapa banyak "ikat" (unit 8) yang dapat dibuat dari nilai sisa saat ini. Akhirnya, kepingSisa := sisa % 8 menghitung sisa akhir setelah semua keping yang mungkin telah dikelompokkan menjadi "ikat". Nilai di kepingSisa ini adalah jumlah keping final yang kurang dari 8 dan tidak dapat dikelompokkan lebih lanjut.

Sebagai langkah terakhir, program melaporkan semua hasil perhitungan ini kepada pengguna. fmt.Printf("%d peti, %d karung, %d ikat, dan %d keping\n", peti, karung, ikat, kepingSisa) digunakan untuk mencetak sebuah string yang diformat. Teks di dalam tanda kutip adalah templat, di mana setiap penanda %d secara berurutan digantikan oleh nilai dari variabel peti, karung, ikat, dan kepingSisa. Karakter \n di akhir memastikan bahwa kursor akan pindah ke baris baru setelah keluaran ditampilkan, sehingga membuat tampilan konsol menjadi rapi.