

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2
MODUL III & IV
FUNGSI & PROSEDUR**



Oleh:

RINDI DELA NUR SAFITRI

2311102332

IF-11-07

**S1 TEKNIK INFORMATIKA
UNIVERSITAS TELKOM PURWOKERTO
2024**

I. DASAR TEORI

Modul 3

Fungsi merupakan satu kesatuan rangkaian instruksi yang memberikan atau menghasilkan suatu nilai dan biasanya memetakan input ke suatu nilai yang lain. Oleh karena itu, fungsi selalu menghasilkan/mengembalikan nilai. Suatu subprogram dikatakan fungsi apabila:

1. Ada deklarasi tipe nilai yang dikembalikan, dan
2. Terdapat kata kunci return dalam badan subprogram.

Maka fungsi digunakan jika suatu nilai biasanya diperlukan, seperti:

- Assignment nilai ke suatu variabel
- Bagian dari ekspresi
- Bagian dari argumen suatu subprogram, dsb.

Karena itu selalu pilih nama fungsi yang menggambarkan nilai, seperti kata benda dan kata sifat. Contoh nama-nama fungsi: median, rerata, nilaiTerbesar, ketemu, selesai,...

Deklarasi fungsi sama dengan prosedur, yaitu berada pada blok yang terpisah dengan program utama.

	Notasi Algoritma
1	function <nama function> (<params>) -> <type>
2	kamus
3	{deklarasi variabel lokal dari fungsi}
4	...
5	algoritma
6	{badan algoritma fungsi}
7	...
8	return <value/variabel>
9	endfunction
	Notasi dalam bahasa Go
10	func <nama function> (<params>) <type> {
11	/* deklarasi variabel lokal dari fungsi */
12	...
13	/* badan algoritma fungsi*/
14	...
15	return <value/variabel>
16	}
17	

Modul 4

Prosedur dapat dianggap sebagai potongan beberapa instruksi program menjadi suatu instruksi baru yang dibuat untuk mengurangi kerumitan dari kode program yang kompleks pada suatu program yang besar. Prosedur akan menghasilkan suatu akibat atau efek langsung pada program ketika dipanggil pada program utama. Suatu subprogram dikatakan prosedur apabila:

1. Tidak ada deklarasi tipe nilai yang dikembalikan, dan
2. Tidak terdapat kata kunci return dalam badan subprogram.

Kedudukannya prosedur sama seperti instruksi dasar yang sudah ada sebelumnya (assignment) dan/atau instruksi yang berasal dari paket (fmt), seperti fmt.Scan dan fmt.Print. Karena itu selalu pilih nama prosedur yang berbentuk kata kerja atau sesuatu yang merepresentasikan proses sebagai nama dari prosedur. Contoh: cetak, hitungRerata, cariNilai, belok, mulai,...

Berikut ini adalah cara penulisan deklarasi prosedur pada notasi Pseudocode dan GoLang.

Notasi Algoritma	
1	procedure <nama procedure> (<params>)
2	kamus
3	{deklarasi variabel lokal dari procedure}
4	...
5	algoritma
6	{badan algoritma procedure}
7	...
8	endprocedure
Notasi dalam bahasa Go	
9	func <nama procedure> <(params)> {
10	/* deklarasi variabel lokal dari procedure */
11	...
12	/* badan algoritma procedure */
13	...
14	}

II. GUIDED

Guided 1

Source Code

```
package main

import (
    "fmt"
)

func volumeTabung(jari_jari, tinggi int) float64 {
    var luasAlas, volume float64
    luasAlas = 3.14 * float64(jari_jari*jari_jari)
    volume = luasAlas * float64(tinggi)
    return volume
}

func main() {
    var r, t int
    var v1, v2 float64
    r = 5
    t = 10

    v1 = volumeTabung(r, t)
    v2 = volumeTabung(r, t) + volumeTabung(15, t)

    fmt.Println("Volume tabung 14 x 100: ", volumeTabung(14,
100))
    fmt.Println("v1: ", v1)
    fmt.Println("v2: ", v2)
}
```

Screenshot Program



The screenshot shows a terminal window with a dark blue background. The top part displays the source code of the Go program, which is identical to the code provided in the 'Source Code' section. The bottom part shows the output of the program after running the command `go run "c:\Users\ACER\Downloads\go\guided1.go"`. The output is as follows:

```
PS C:\Users\ACER> go run "c:\Users\ACER\Downloads\go\guided1.go"
Volume tabung 14 x 100:  61544.00000000001
v1:  785
v2:  7850
```

Penjelasan

Program di atas merupakan implementasi sederhana dalam bahasa Go yang menghitung volume sebuah tabung. Fungsi `volumeTabung` menerima dua parameter, yaitu `jari_jari` dan `tinggi`, yang bertipe data integer, lalu mengembalikan volume tabung sebagai tipe data `float64`. Volume dihitung dengan cara mengalikan luas alas tabung (yang berbentuk lingkaran dengan rumus $\pi \times \text{jari-jari}^2$) dengan tinggi tabung. Fungsi ini menggunakan nilai π sebagai 3.14 untuk mempermudah perhitungan.

Di dalam fungsi `main`, beberapa variabel seperti `r`, `t`, `v1`, dan `v2` diinisialisasi. Nilai `r` dan `t` masing-masing diberikan 5 dan 10, yang kemudian digunakan untuk menghitung volume tabung melalui fungsi `volumeTabung`. Program juga menghitung dan mencetak hasil dari beberapa volume tabung, seperti tabung dengan jari-jari 14 dan tinggi 100, serta nilai volume `v1` dan `v2`. Output akhirnya akan menampilkan volume-volume yang sudah dihitung tersebut di terminal.

Guided 2

Source Code

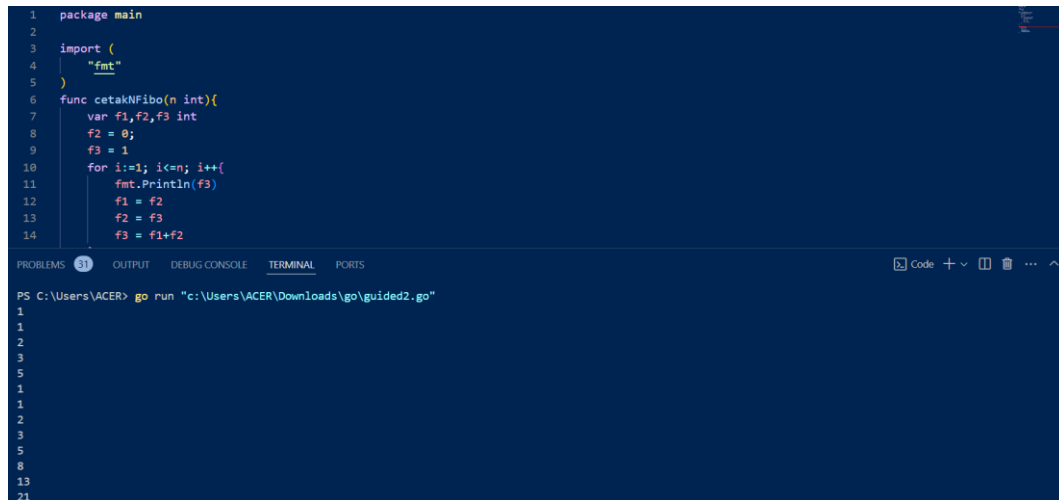
```
package main

import (
    "fmt"
)

func cetakNFibo(n int){
    var f1,f2,f3 int
    f2 = 0;
    f3 = 1
    for i:=1; i<=n; i++){
        fmt.Println(f3)
        f1 = f2
        f2 = f3
        f3 = f1+f2
    }
}

func main(){
    var x int
    x = 5
    cetakNFibo(x)
    cetakNFibo(100)
}
```

Screenshot Program



```
1 package main
2
3 import (
4     "fmt"
5 )
6 func cetakNFibo(n int){
7     var f1,f2,f3 int
8     f2 = 0;
9     f3 = 1
10    for i:=1; i<=n; i++){
11        fmt.Println(f3)
12        f1 = f2
13        f2 = f3
14        f3 = f1+f2
15    }
16 }
17
18 func main(){
19     cetakNFibo(5)
20     cetakNFibo(100)
21 }
```

PS C:\Users\ACER> go run "c:\Users\ACER\Downloads\go\guided2.go"

1
1
2
3
5
1
1
2
3
5
8
13
21

Penjelasan

Program di atas merupakan implementasi sederhana untuk mencetak deret Fibonacci hingga sejumlah elemen tertentu dalam bahasa Go. Deret Fibonacci adalah deret bilangan di mana setiap bilangan setelah dua bilangan pertama merupakan penjumlahan dari dua bilangan sebelumnya. Fungsi `cetakNFibo(n int)` bertugas mencetak `n` elemen pertama dari deret Fibonacci. Di dalam fungsi ini, tiga variabel digunakan: `f1`, `f2`, dan `f3` untuk menyimpan nilai-nilai dari dua bilangan sebelumnya dan hasil penjumlahan mereka.

Pada fungsi `cetakNFibo`, `f2` diinisialisasi dengan nilai 0 (bilangan pertama dalam Fibonacci) dan `f3` dengan nilai 1 (bilangan kedua dalam Fibonacci). Di dalam loop `for`, program mencetak nilai `f3`, lalu memperbarui nilai `f1`, `f2`, dan `f3` dengan pergeseran maju di setiap iterasi, sehingga membentuk urutan Fibonacci. Pada fungsi `main`, program memanggil `cetakNFibo` dua kali, pertama untuk mencetak 5 bilangan Fibonacci dan kemudian untuk mencetak 100 bilangan Fibonacci.

III. UNGUIDED

Modul 3

Unguided 1

1. Minggu ini, mahasiswa Fakultas Informatika mendapatkan tugas dari mata kuliah matematika diskrit untuk mempelajari kombinasi dan permutasi. Jonas salah seorang mahasiswa, iseng untuk mengimplementasikannya ke dalam suatu program. Oleh karena itu bersediakah kalian membantu Jonas? (tidak tentunya ya :p)

Masukan terdiri dari empat buah bilangan asli a, b, c, dan d yang dipisahkan oleh spasi, dengan syarat $a \leq c$ dan $b \geq d$.

Keluaran terdiri dari dua baris. Baris pertama adalah hasil permutasi dan kombinasi a terhadap c, sedangkan baris kedua adalah hasil permutasi dan kombinasi b terhadap d.

Catatan: permutasi (P) dan kombinasi (C) dari n terhadap r ($n \geq r$) dapat dihitung dengan menggunakan persamaan berikut!

$$P(n, r) = \frac{n!}{(n-r)!}, \text{ sedangkan } C(n, r) = \frac{n!}{r!(n-r)!}$$

Contoh

No	Masukan	Keluaran	Penjelasan
1	5 10 3 10	60 10 3628800 1	$P(5, 3) = 5! / 2! = 120 / 2 = 60$ $C(5, 3) = 5! / (3! \times 2!) = 120 / 12 = 10$ $P(10, 10) = 10! / 0! = 3628800 / 1 = 3628800$ $C(10, 10) = 10! / (10! \times 0!) = 10! / 10! = 1$
2	8 0 2 0	56 28 1 1	

Selesaikan program tersebut dengan memanfaatkan subprogram yang diberikan berikut ini!

```
function factorial(n: integer) → integer
{mengembalikan nilai faktorial dari n}

function permutation(n,r : integer) → integer
{Mengembalikan hasil n permutasi r, dan n >= r}

function combination(n,r : integer) → integer
{Mengembalikan hasil n kombinasi r, dan n >= r}
```


Source Code

```
package main

import "fmt"

func factorial(n int) int {
    if n == 0 {
        return 1
    }
    return n * factorial(n-1)
}

func permutation(n, r int) int {
    return factorial(n) / factorial(n-r)
}

func combination(n, r int) int {
    return factorial(n) / (factorial(r) * factorial(n-r))
}

func main() {
    var a, b, c, d int

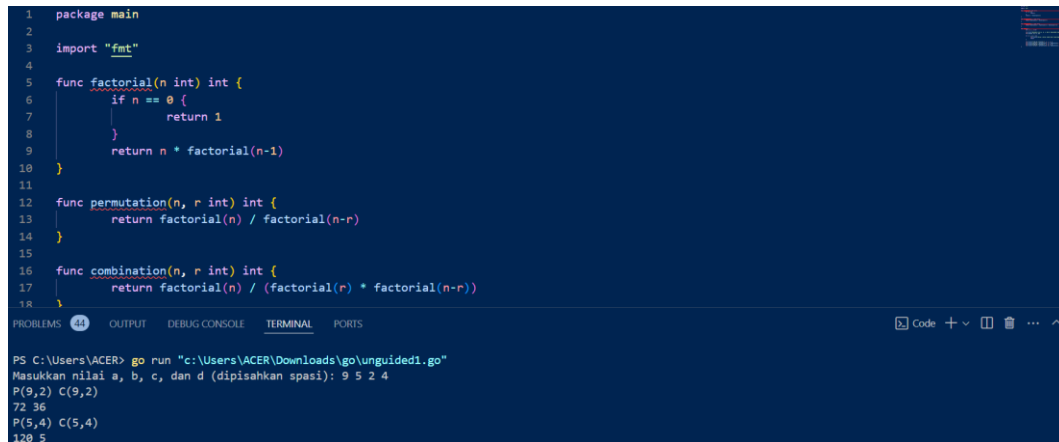
    fmt.Print("Masukkan nilai a, b, c, dan d (dipisahkan spasi): ")
    fmt.Scan(&a, &b, &c, &d)

    if a < c || b < d {
        fmt.Println("Nilai a harus lebih besar sama dengan c, dan b harus lebih besar sama dengan d.")
        return
    }

    fmt.Printf("P(%d,%d) C(%d,%d)\n", a, c, a, c)
    fmt.Printf("%d %d\n", permutation(a, c), combination(a, c))

    fmt.Printf("P(%d,%d) C(%d,%d)\n", b, d, b, d)
    fmt.Printf("%d %d\n", permutation(b, d), combination(b, d))
}
```

Screenshot Program



```
1 package main
2
3 import "fmt"
4
5 func factorial(n int) int {
6     if n == 0 {
7         return 1
8     }
9     return n * factorial(n-1)
10 }
11
12 func permutation(n, r int) int {
13     return factorial(n) / factorial(n-r)
14 }
15
16 func combination(n, r int) int {
17     return factorial(n) / (factorial(r) * factorial(n-r))
18 }
19
20 func main() {
21     // ... (main function logic) ...
22 }
```

PS C:\Users\ACER> go run "c:\Users\ACER\Downloads\go\unguided1.go"

Masukkan nilai a, b, c, dan d (dipisahkan spasi): 9 5 2 4

P(9,2) C(9,2)

72 36

P(5,4) C(5,4)

120 5

Penjelasan

Program di atas merupakan implementasi perhitungan. Fungsi-fungsi yang digunakan dalam program ini adalah `factorial`, `permutation`, dan `combination`.

1. Fungsi `factorial(n int) int`: Fungsi ini menggunakan rekursi untuk menghitung faktorial dari bilangan n . Jika n sama dengan 0, maka akan mengembalikan nilai 1 (karena $0! = 1$). Jika tidak, fungsi tersebut mengembalikan hasil perkalian dari n dengan faktorial dari $n-1$.
2. Fungsi `permutation(n, r int) int`: Fungsi ini menghitung permutasi dari n objek yang diambil r secara berurutan, menggunakan rumus $P(n, r) = n! / (n-r)!$.
3. Fungsi `combination(n, r int) int`: Fungsi ini menghitung kombinasi dari n objek yang diambil r tanpa memperhatikan urutan, menggunakan rumus $C(n, r) = n! / (r! * (n-r)!)$.

Di dalam `main`, pengguna diminta untuk memasukkan empat nilai a , b , c , dan d . Program kemudian memverifikasi apakah nilai $a \geq c$ dan $b \geq d$. Jika kondisi ini tidak terpenuhi, program menampilkan pesan kesalahan. Jika kondisi terpenuhi, program akan mencetak hasil permutasi dan kombinasi untuk pasangan (a, c) dan (b, d) sesuai input.

Unguided 2

2. Diberikan tiga buah fungsi matematika yaitu $f(x) = x^2$, $g(x) = x - 2$ dan $h(x) = x + 1$.

Fungsi komposisi $(f \circ g \circ h)(x)$ artinya adalah $f(g(h(x)))$. Tuliskan $f(x)$, $g(x)$ dan $h(x)$ dalam bentuk function.

Masukan terdiri dari sebuah bilangan bulat a , b dan c yang dipisahkan oleh spasi.

Keluaran terdiri dari tiga baris. Baris pertama adalah $(f \circ g \circ h)(a)$, baris kedua $(g \circ h \circ f)(b)$, dan baris ketiga adalah $(h \circ f \circ g)(c)$!

Contoh

No	Masukan	Keluaran	Penjelasan
1	7 2 10	36 3 65	$(f \circ g \circ h)(7) = 36$ $(g \circ h \circ f)(2) = 3$ $(h \circ f \circ g)(10) = 65$
2	5 5 5	16 24 10	$(f \circ g \circ h)(5) = 16$ $(g \circ h \circ f)(5) = 24$ $(h \circ f \circ g)(5) = 10$
3	3 8 4	4 63 5	$(f \circ g \circ h)(5) = 4$ $(g \circ h \circ f)(5) = 63$ $(h \circ f \circ g)(5) = 5$

Source Code

```
package main

import "fmt"

func f(x int) int {
    return x * x
}

func g(x int) int {
    return x - 2
}

func h(x int) int {
    return x + 1
}

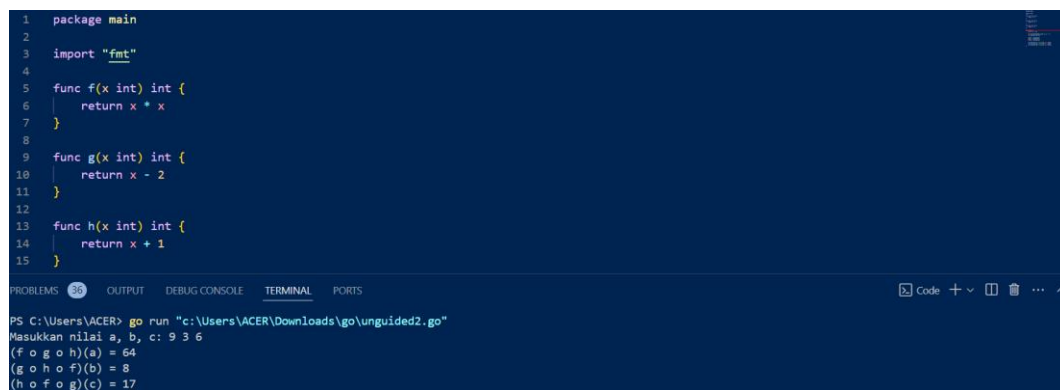
func main() {
    var a, b, c int

    fmt.Print("Masukkan nilai a, b, c: ")
    fmt.Scan(&a, &b, &c)

    fogoh := f(g(h(a)))
    gohof := g(h(f(b)))
    hofog := h(f(g(c)))

    fmt.Println("(f o g o h)(a) =", fogoh)
    fmt.Println("(g o h o f)(b) =", gohof)
    fmt.Println("(h o f o g)(c) =", hofog)
}
```

Screenshot Program



The screenshot shows a terminal window with a dark blue background. The top part displays the source code of a Go program, which defines three functions: `f` (multiplication), `g` (subtraction by 2), and `h` (addition by 1). The `main` function prompts the user to input three integers `a`, `b`, and `c`. Below the code, the terminal output shows the results of the function compositions for the input values 9, 3, and 6. The output is as follows:

```
1 package main
2
3 import "fmt"
4
5 func f(x int) int {
6     return x * x
7 }
8
9 func g(x int) int {
10    return x - 2
11 }
12
13 func h(x int) int {
14    return x + 1
15 }

PS C:\Users\ACER> go run "c:\Users\ACER\Downloads\go\unguided2.go"
Masukkan nilai a, b, c: 9 3 6
(f o g o h)(a) = 64
(g o h o f)(b) = 8
(h o f o g)(c) = 17
```

Penjelasan

Program di atas mendefinisikan tiga fungsi: $f(x)$, $g(x)$, dan $h(x)$. Fungsi $f(x)$ mengembalikan nilai kuadrat dari x , fungsi $g(x)$ mengurangi x dengan 2, dan fungsi $h(x)$ menambah x dengan 1. Pada fungsi `main()`, program meminta input tiga bilangan integer a , b , dan c dari pengguna. Kemudian, program menghitung dan menyusun beberapa komposisi fungsi, seperti $f(g(h(a)))$, $g(h(f(b)))$, dan $h(f(g(c)))$, yang dihitung secara berurutan dari dalam ke luar.

Setelah perhitungan selesai, hasil dari komposisi fungsi ini dicetak ke layar. Setiap komposisi menggambarkan bagaimana fungsi diterapkan satu sama lain pada variabel input yang diberikan pengguna. Sebagai contoh, $(f \circ g \circ h)(a)$ mengartikan bahwa fungsi h diterapkan pada a , kemudian hasilnya diteruskan ke fungsi g , dan hasil akhirnya digunakan oleh fungsi f .

Unguided 3

3. **[Lingkaran]** Suatu lingkaran didefinisikan dengan koordinat titik pusat (cx, cy) dengan radius r . Apabila diberikan dua buah lingkaran, maka tentukan posisi sebuah titik sembarang (x, y) berdasarkan dua lingkaran tersebut.

Masukan terdiri dari beberapa tiga baris. Baris pertama dan kedua adalah koordinat titik pusat dan radius dari lingkaran 1 dan lingkaran 2, sedangkan baris ketiga adalah koordinat titik sembarang. Asumsi sumbu x dan y dari semua titik dan juga radius direpresentasikan dengan bilangan bulat.

Keluaran berupa string yang menyatakan posisi titik **"Titik di dalam lingkaran 1 dan 2"**, **"Titik di dalam lingkaran 1"**, **"Titik di dalam lingkaran 2"**, atau **"Titik di luar lingkaran 1 dan 2"**.

Contoh

No	Masukan	Keluaran
1	1 1 5 8 8 4 2 2	Titik di dalam lingkaran 1
2	1 2 3	Titik di dalam lingkaran 2

	4 5 6 7 8	
3	5 10 15 -15 4 20 0 0	Titik di dalam lingkaran 1 dan 2
4	1 1 5 8 8 4 15 20	Titik di luar lingkaran 1 dan 2

Fungsi untuk menghitung jarak titik (a, b) dan (c, d) dimana rumus jarak adalah:

$$jarak = \sqrt{(a - c)^2 + (b - d)^2}$$

dan juga fungsi untuk menentukan posisi sebuah titik sembarang berada di dalam suatu lingkaran atau tidak.

```
function jarak(a,b,c,d : real) -> real
{Mengembalikan jarak antara titik (a,b) dan titik (c,d)}

function didalam(cx,cy,r,x,y : real) -> boolean
{Mengembalikan true apabila titik (x,y) berada di dalam lingkaran yang
memiliki titik pusat (cx,cy) dan radius r}
```

Catatan: Lihat paket **math** dalam lampiran untuk menggunakan fungsi **math.Sqrt()** untuk menghitung akar kuadrat.

Source Code

```
package main

import (
    "fmt"
    "math"
)

func jarak(x1, y1, x2, y2 float64) float64 {
    return math.Sqrt(math.Pow(x2-x1, 2) + math.Pow(y2-y1,
2))
}

func dalamLingkaran(cx, cy, r, x, y float64) bool {
    return jarak(cx, cy, x, y) <= r
}

func main() {

    var cx1, cy1, r1 float64
    fmt.Print("Masukkan cx1, cy1, r1 (Lingkaran 1): ")
    fmt.Scan(&cx1, &cy1, &r1)

    var cx2, cy2, r2 float64
    fmt.Print("Masukkan cx2, cy2, r2 (Lingkaran 2): ")
    fmt.Scan(&cx2, &cy2, &r2)

    var x, y float64
    fmt.Print("Masukkan koordinat titik (x, y): ")
    fmt.Scan(&x, &y)

    dalam1 := dalamLingkaran(cx1, cy1, r1, x, y)
    dalam2 := dalamLingkaran(cx2, cy2, r2, x, y)

    if dalam1 && dalam2 {
        fmt.Println("Titik di dalam lingkaran 1 dan 2")
    } else if dalam1 {
        fmt.Println("Titik di dalam lingkaran 1")
    } else if dalam2 {
        fmt.Println("Titik di dalam lingkaran 2")
    } else {
        fmt.Println("Titik di luar lingkaran 1 dan 2")
    }
}
```

Screenshot Program



```
1 package main
2
3 import (
4     "fmt"
5     "math"
6 )
7
8 func jarak(x1, y1, x2, y2 float64) float64 {
9     return math.Sqrt(math.Pow(x2-x1, 2) + math.Pow(y2-y1, 2))
10 }
11
12 func dalamLingkaran(cx, cy, r, x, y float64) bool {
13     return jarak(cx, cy, x, y) <= r
14 }
15
16 func main() {
17     var cx1, cy1, r1 float64
18
19     PS C:\Users\ACER> go run "c:\Users\ACER\Downloads\go\unguided3.go"
20 Masukkan cx1, cy1, r1 (Lingkaran 1): 25 38 9
21 Masukkan cx2, cy2, r2 (Lingkaran 2): 21 14 10
22 Masukkan koordinat titik (x, y): 7 9
23 Titik di luar lingkaran 1 dan 2
```

Penjelasan

Program di atas bertujuan untuk memeriksa apakah sebuah titik berada di dalam dua lingkaran atau tidak. Fungsi `jarak` menghitung jarak antara dua titik menggunakan rumus jarak Euclidean, sedangkan fungsi `dalamLingkaran` memeriksa apakah titik tertentu berada di dalam lingkaran dengan cara membandingkan jarak antara titik pusat lingkaran dengan titik yang diperiksa terhadap radius lingkaran. Jika jarak tersebut kurang dari atau sama dengan radius, maka titik berada di dalam lingkaran.

Pada bagian utama program, pengguna diminta untuk memasukkan koordinat pusat dan radius dari dua lingkaran, serta koordinat titik yang akan diuji. Program kemudian menggunakan fungsi `dalamLingkaran` untuk memeriksa apakah titik tersebut berada di dalam salah satu atau kedua lingkaran. Hasil pemeriksaan ditampilkan dengan pesan yang menyatakan apakah titik berada di dalam lingkaran pertama, lingkaran kedua, keduanya, atau di luar kedua lingkaran.

Modul 4

Unguided 4

1. Minggu ini, mahasiswa Fakultas Informatika mendapatkan tugas dari mata kuliah matematika diskrit untuk mempelajari kombinasi dan permutasi. Jonas salah seorang mahasiswa, iseng untuk mengimplementasikannya ke dalam suatu program. Oleh karena itu bersedialah kalian membantu Jonas? (tidak tentunya ya :p)

Contoh

No	Masukan	Keluaran	Penjelasan
1	5 10 3 10	60 10 3628800 1	$P(5,3) = 5!/2! = 120/2 = 60$ $C(5,3) = 5!/(3! \times 2!) = 120/12 = 10$ $P(10,10) = 10!/0! = 3628800/1 = 3628800$ $C(10,10) = 10!/(10! \times 0!) = 10!/10! = 1$
2	8 0 2 0	56 28 1 1	

Selesaikan program tersebut dengan memanfaatkan prosedur yang diberikan berikut ini!

```

procedure factorial(in n: integer, in/out hasil:integer)
{I.S. terdefinisi bilangan bulat positif n
 F.S. hasil berisi nilai faktorial dari n}

procedure permutation(in n,r : integer, in/out hasil:integer)
{I.S. terdefinisi bilangan bulat positif n dan r, dan n >= r
 F.S. hasil berisi nilai dari n permutasi r}

procedure combination(in n,r : integer, in/out hasil:integer)
{I.S. terdefinisi bilangan bulat positif n dan r, dan n >= r
 F.S. hasil berisi nilai dari n kombinasi r}
  
```

Source Code

```
package main

import "fmt"

func factorial(n int, result *int) {
    if n == 0 {
        *result = 1
    } else {
        *result = n
        temp := 1
        factorial(n-1, &temp)
        *result *= temp
    }
}

func permutation(n, r int, result *int) {
    var factN, factNR int
    factorial(n, &factN)
    factorial(n-r, &factNR)
    *result = factN / factNR
}

func combination(n, r int, result *int) {
    var factN, factR, factNR int
    factorial(n, &factN)
    factorial(r, &factR)
    factorial(n-r, &factNR)
    *result = factN / (factR * factNR)
}

func main() {
    var a, b, c, d int

    fmt.Print("Masukkan nilai a, b, c, dan d (dipisahkan spasi): ")
    fmt.Scan(&a, &b, &c, &d)

    if a < c || b < d {
        fmt.Println("Nilai a harus lebih besar sama dengan c, dan b harus lebih besar sama dengan d.")
        return
    }
}
```

```

var permAC, combAC, permBD, combBD int

    permutation(a, c, &permAC)
    combination(a, c, &combAC)
    fmt.Printf("P(%d,%d) C(%d,%d)\n", a, c, a, c)
    fmt.Printf("%d %d\n", permAC, combAC)

    permutation(b, d, &permBD)
    combination(b, d, &combBD)
    fmt.Printf("P(%d,%d) C(%d,%d)\n", b, d, b, d)
    fmt.Printf("%d %d\n", permBD, combBD)
}

```

Screenshot Program



```

1 package main
2
3 import "fmt"
4
5 func factorial(n int, result *int) {
6     if n == 0 {
7         *result = 1
8     } else {
9         *result = n
10        temp := 1
11        factorial(n-1, &temp)
12        *result *= temp
13    }
14 }
15
16 func permutation(n, r int, result *int) {
17     var factN, factNR int
18     factorial(n, &factN)

```

PS C:\Users\ACER> go run "c:\Users\ACER\Downloads\go\unguided4.go"
 Masukkan nilai a, b, c, dan d (dipisahkan spasi): 8 6 2 4
 P(8,2) C(8,2)
 56 28
 P(6,4) C(6,4)
 360 15

Penjelasan

Program di atas mendefinisikan beberapa prosedur untuk menghitung faktorial, permutasi, dan kombinasi menggunakan rekursi serta pointer untuk menyimpan hasilnya. Prosedur `factorial` mengambil dua parameter, yaitu nilai integer `n` dan pointer ke `result`. Fungsi ini bekerja secara rekursif untuk menghitung nilai faktorial, dan hasilnya disimpan di alamat yang ditunjukkan oleh pointer `result`. Prosedur `permutation` dan `combination` menggunakan hasil dari fungsi `factorial` untuk menghitung permutasi $P(n, r)$ dan kombinasi $C(n, r)$, masing-masing dengan menggunakan rumus matematis yang sesuai.

Fungsi `main` meminta input dari pengguna, yaitu nilai `a`, `b`, `c`, dan `d`, kemudian memverifikasi bahwa `a >= c` dan `b >= d`. Setelah verifikasi, fungsi memanggil prosedur `permutation` dan `combination` untuk menghitung permutasi dan kombinasi antara nilai `a` dan `c`, serta `b` dan `d`. Hasilnya kemudian ditampilkan.

ke layar. Program ini memberikan output berupa nilai permutasi dan kombinasi untuk dua set pasangan angka.

Unguided 5

2. Kompetisi pemrograman tingkat nasional berlangsung ketat. Setiap peserta diberikan 8 soal yang harus dapat diselesaikan dalam waktu 5 jam saja. Peserta yang berhasil menyelesaikan soal paling banyak dalam waktu paling singkat adalah pemenangnya.

Buat program **gema** yang mencari pemenang dari daftar peserta yang diberikan. Program harus dibuat modular, yaitu dengan membuat prosedur `hitungSkor` yang mengembalikan total soal dan total skor yang dikerjakan oleh seorang peserta, melalui parameter formal. Pembacaan nama peserta dilakukan di program utama, sedangkan waktu pengerjaan dibaca di dalam prosedur.

prosedure `hitungSkor(in/out soal, skor : integer)`

Setiap baris **masukan** dimulai dengan satu string nama peserta tersebut diikuti dengan adalah 8 integer yang menyatakan berapa lama (dalam menit) peserta tersebut menyelesaikan soal. Jika tidak berhasil atau tidak mengirimkan jawaban maka otomatis dianggap menyelesaikan dalam waktu 5 jam 1 menit (301 menit).

Satu baris **keluaran** berisi nama pemenang, jumlah soal yang diselesaikan, dan nilai yang diperoleh. Nilai adalah total waktu yang dibutuhkan untuk menyelesaikan soal yang berhasil diselesaikan.

No	Masukan	Keluaran
1	Astuti 20 50 301 301 61 71 75 10 Bertha 25 47 301 26 50 60 65 21 Selesai	Bertha 7 294

Keterangan:

Astuti menyelesaikan 6 soal dalam waktu 287 menit, sedangkan Bertha 7 soal dalam waktu 294 menit. Karena Bertha menyelesaikan lebih banyak, maka Bertha menang. Jika keduanya menyelesaikan sama banyak, maka pemenang adalah yang menyelesaikan dengan total waktu paling kecil.

Source Code

```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strings"
)

func hitungSkor(input string) (soal int, skor int) {
    parts := strings.Fields(input)
    soal = 0
    skor = 0

    for i := 1; i < len(parts); i++ {
        var waktu int
        fmt.Sscan(parts[i], &waktu)

        if waktu <= 300 {
            soal++
            skor += waktu
        }
    }

    return soal, skor
}

func main() {
    scanner := bufio.NewScanner(os.Stdin)
    var inputData []string

    fmt.Println("Masukkan daftar peserta:")

    for {
        scanner.Scan()
        line := scanner.Text()

        if strings.ToLower(line) == "selesai" {
            break
        }

        inputData = append(inputData, line)
    }
}
```

```

var pemenang string
var maxSoal, minSkor int
maxSoal = 0
minSkor = 1000000

for _, data := range inputData {
    soal, skor := hitungSkor(data)
    nama := strings.Fields(data)[0]

    if soal > maxSoal || (soal == maxSoal && skor <
minSkor) {
        pemenang = nama
        maxSoal = soal
        minSkor = skor
    }
}

fmt.Println("\nHasil:")
fmt.Printf("%s %d %d\n", pemenang, maxSoal, minSkor)
}

```

Screenshot Program

```

1 package main
2
3 import (
4     "bufio"
5     "fmt"
6     "os"
7     "strings"
8 )
9
10 func hitungSkor(input string) (soal int, skor int) {
11     parts := strings.Fields(input)
12     soal = 0
13     skor = 0
14
15     for i := 1; i < len(parts); i++ {
16         var waktu int
17         fmt.Scan(parts[i], &waktu)
18     }
19
20     soal = parts[0]
21     for i := 1; i < len(parts); i++ {
22         skor += waktu
23     }
24 }
25
26 func main() {
27     input := "Dela 30 56 78 209 98 110 46 138"
28     soal, skor := hitungSkor(input)
29     fmt.Println("Hasil:")
30     fmt.Printf("%s %d %d\n", input, soal, skor)
31 }

```

PS C:\Users\ACER> go run "c:\Users\ACER\Downloads\go\unguided5.go"

Masukkan daftar peserta:

Dela 30 56 78 209 98 110 46 138

Ayyub 302 28 47 87 199 42 202 35

selesai

Hasil:

Dela 8 765

Penjelasan

Program di atas bertujuan untuk menghitung skor dan menentukan pemenang berdasarkan waktu yang dihabiskan oleh peserta dalam suatu kegiatan. Fungsi `hitungSkor` menerima sebuah string input yang berisi nama peserta diikuti oleh sejumlah waktu yang digunakan untuk menyelesaikan soal. Di dalam fungsi ini, program memisahkan input menjadi bagian-bagian dan mengiterasi melalui waktu yang diberikan. Jika waktu yang dihabiskan peserta kurang dari atau sama dengan

300 detik, jumlah soal (`soal`) yang diselesaikan dan total skor (`skor`) peserta akan diupdate.

Dalam fungsi `main`, program menggunakan `bufio.Scanner` untuk membaca input dari pengguna, memungkinkan mereka untuk memasukkan nama peserta beserta waktu yang dihabiskan hingga kata "selesai" dimasukkan. Setelah itu, program akan memproses setiap input dengan memanggil fungsi `hitungSkor` untuk menentukan siapa yang menyelesaikan jumlah soal terbanyak dengan skor terendah. Pemenang ditentukan berdasarkan kriteria tersebut, dan hasilnya dicetak ke layar. Program ini menunjukkan penggunaan prosedur dalam pengolahan data dan pengambilan keputusan berdasarkan kriteria yang telah ditentukan.

Unguided 6

3.

Skiena dan Revilla dalam *Programming Challenges* mendefinisikan sebuah deret bilangan. Deret dimulai dengan sebuah bilangan bulat n . Jika bilangan n saat itu genap, maka suku berikutnya adalah $\frac{1}{2}n$, tetapi jika ganjil maka suku berikutnya bernilai $3n+1$. Rumus yang sama digunakan terus menerus untuk mencari suku berikutnya. Deret berakhir ketika suku terakhir bernilai 1. Sebagai contoh jika dimulai dengan $n=22$, maka deret bilangan yang diperoleh adalah:

22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

Untuk suku awal sampai dengan 1000000, diketahui deret selalu mencapai suku dengan nilai 1.

Buat program **skiena** yang akan mencetak setiap suku dari deret yang dijelaskan di atas untuk nilai suku awal yang diberikan. Pencetakan deret harus dibuat dalam prosedur `cetakDeret` yang mempunyai 1 parameter formal, yaitu nilai dari suku awal.

prosedure `cetakDeret(in n : integer)`

Masukan berupa satu bilangan integer positif yang lebih kecil dari 1000000.

Keluaran terdiri dari satu baris saja. Setiap suku dari deret tersebut dicetak dalam baris yang dan dipisahkan oleh sebuah spasi.

No	Masukan	Keluaran
1	22	22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

Source Code

```
package main

import (
    "fmt"
)

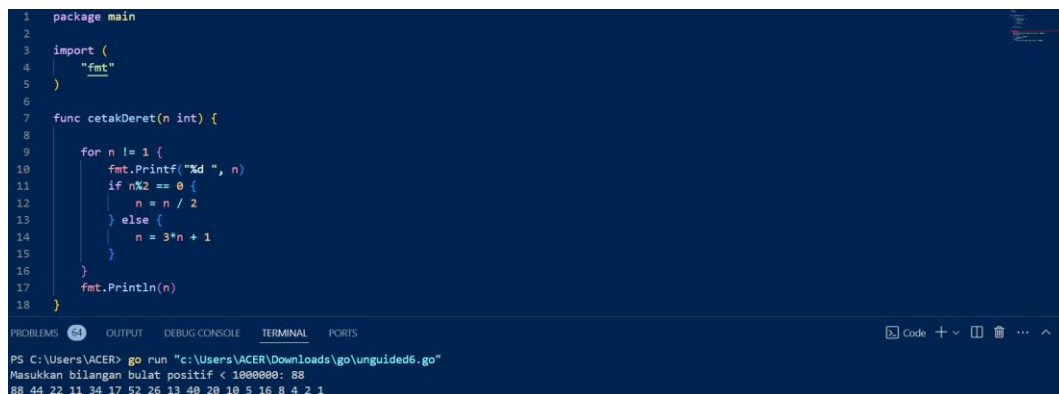
func cetakDeret(n int) {

    for n != 1 {
        fmt.Printf("%d ", n)
        if n%2 == 0 {
            n = n / 2
        } else {
            n = 3*n + 1
        }
    }
    fmt.Println(n)
}

func main() {
    var n int
    fmt.Print("Masukkan bilangan bulat positif < 1000000: ")
    fmt.Scan(&n)

    if n > 0 && n < 1000000 {
        cetakDeret(n)
    } else {
        fmt.Println("Bilangan harus positif < 1000000")
    }
}
```

Screenshot Program



```
1 package main
2
3 import (
4     "fmt"
5 )
6
7 func cetakDeret(n int) {
8
9     for n != 1 {
10         fmt.Printf("%d ", n)
11         if n%2 == 0 {
12             n = n / 2
13         } else {
14             n = 3*n + 1
15         }
16     }
17     fmt.Println(n)
18 }
19
20 func main() {
21     var n int
22     fmt.Print("Masukkan bilangan bulat positif < 1000000: ")
23     fmt.Scan(&n)
24
25     if n > 0 && n < 1000000 {
26         cetakDeret(n)
27     } else {
28         fmt.Println("Bilangan harus positif < 1000000")
29     }
30 }
```

PS C:\Users\ACER> go run "c:\Users\ACER\Downloads\go\unguided6.go"

Masukkan bilangan bulat positif < 1000000: 88

88 44 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

Penjelasan

Program di atas bertujuan untuk mencetak deret bilangan sesuai dengan aturan yang dikenal sebagai Conjecture Collatz. Prosedur `cetakDeret` menerima satu parameter integer `n`, yang merupakan bilangan bulat positif. Di dalam prosedur ini, terdapat sebuah loop yang akan terus berjalan selama `n` tidak sama dengan 1. Selama iterasi, program mencetak nilai `n` saat ini, lalu memeriksa apakah `n` genap atau ganjil. Jika `n` genap, maka nilainya dibagi dua; jika ganjil, nilainya dihitung dengan rumus $3n + 1$. Proses ini berlanjut hingga `n` mencapai 1, di mana program juga mencetak 1 sebelum mengakhiri proses.

Fungsi `main` bertanggung jawab untuk mengambil input dari pengguna. Pertama, program meminta pengguna untuk memasukkan bilangan bulat positif yang lebih kecil dari 1.000.000. Setelah pengguna memberikan input, program memeriksa apakah nilai tersebut berada dalam rentang yang valid (positif dan kurang dari 1.000.000). Jika input valid, prosedur `cetakDeret` dipanggil dengan parameter `n` yang dimasukkan oleh pengguna, dan deret Collatz dicetak. Jika input tidak valid, program menampilkan pesan kesalahan yang menjelaskan bahwa bilangan yang dimasukkan harus positif dan kurang dari 1.000.000.