

**LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK**

**MODUL V
REKURSIF**



Oleh:

OKTAVANIA AYU RAHMADANTY

2311102240

S1IF 11 02

**S1 TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

2024

I. DASAR TEORI

A. Pengantar Rekursif

Pada modul-modul sebelumnya sudah dijelaskan bahwa suatu subprogram baik fungsi atau prosedur bisa memanggil subprogram lainnya. Hal ini tidak menutup kemungkinan bahwa subprogram yang dipanggil adalah dirinya sendiri. Dalam pemrograman teknik ini dikenal dengan istilah rekursif.

Rekursif secara sederhana dapat diartikan sebagai cara menyelesaikan suatu masalah dengan cara menyelesaikan sub-masalah yang identik dari masalah utama. Sebagai contoh perhatikan prosedur cetak berikut ini!

	Notasi Algoritma	Notasi dalam bahasa GO
1	procedure cetak(in x:integer)	func cetak(x int){
2	algoritma	fmt.Println(x)
3	output(x)	cetak(x+1)
4	cetak(x+1)	}
5	endprocedure	

Apabila diperhatikan subprogram cetak() di atas, terlihat pada baris ke-4 terdapat pemanggilan subprogram cetak() kembali. Misalnya apabila kita eksekusi perintah cetak(5) maka akan menampilkan angka 5 6 7 8 9...dst tanpa henti. Artinya setiap pemanggilan subprogram cetak() nilai x akan selalu bertambah 1 (increment by one) secara terus menerus tanpa henti.

```

1 package main
2 import "fmt"
3 func main(){
4     cetak(5)
5 }
6 func cetak(x int){
7     fmt.Println(x)
8     cetak(x+1)
9 }

```

D:\DEV\DEMO>go build contoh.go

D:\DEV\DEMO>contoh.exe

```

5
6
7
8
9
10
11
12
13
...

```

Oleh karena itu biasanya ditambahkan struktur control percabangan (if-then) untuk menghentikan proses rekursif ini. Kondisi ini disebut juga dengan base-case, artinya apabila kondisi base-case bernilai true maka proses rekursif akan berhenti. Sebagai contoh misalnya base case adalah Ketika x bernilai 10 atau $x == 10$, maka tidak perlu dilakukan rekursif.

```

1 procedure cetak(in x:integer)
2 algoritma
3     if x == 10 then
4         output(x)
5     else
6         output(x)
7         cetak(x+1)
8     endif
9 endprocedure

```

Apabila diperhatikan pada baris ke-3 di program di atas, kita telah menambahkan base-case seperti penjelasan sebelumnya. Selanjutnya pada bagian aksi dari else di baris ke-6 dan ke-7 kita namakan recursive-case atau kasus pemanggilan dirinya sendiri tersebut terjadi. Kondisi dari recursive-case ini adalah negasi dari kondisi base-case atau ketika nilai $x \neq 10$.

```

1 package main
2 import "fmt"
3 func main(){
4     cetak(5)
5 }
6 func cetak(x int){
7     if x == 10 {
8         fmt.Println(x)
9     }else{
10        fmt.Println(x)
11        cetak(x+1)
12    }
13 }

```

```

D:\DEV\DEMO>go build contoh.go
D:\DEV\DEMO>contoh.exe

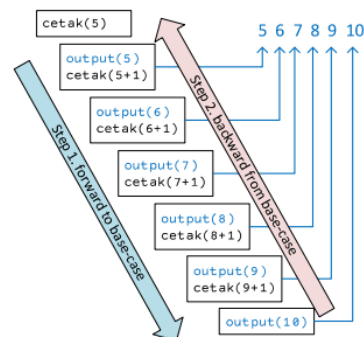
```

```

5
6
7
8
9
10

```

Apabila program di atas ini dijalankan maka akan tampil angka 5 6 7 8 9 10. Terlihat bahwa proses rekursif berhasil dihentikan ketika `x == 10`.



Gambar 1. Ilustrasi proses forward dan backward pada saat rekursif.

Pada Gambar 2 memperlihatkan saat subprogram dipanggil secara rekursif, maka subprogram akan terus melakukan pemanggilan (forward) hingga terhenti pada saat kondisi base-case terpenuhi atau true. Setelah itu akan terjadi proses backward atau Kembali ke subprogramnya yang sebelumnya. Artinya setelah semua intruksi cetak(10) selesai dieksekusi, maka program akan Kembali ke cetak(9) yang memanggil cetak (10) tersebut. Begitu seterusnya hingga Kembali ke cetak(5).

Perhatikan modifikasi program di atas dengan menukar posisi baris 10 dan 11, mengakibatkan Ketika program dijalankan maka akan menampilkan 10 9 8 7 6 5. Kenapa bisa demikian? Pahami proses backward pada gambar 2.

```
1 package main
2 import "fmt"
3 func main(){
4     cetak(5)
5 }
6 func cetak(x int){
7     if x == 10 {
8         fmt.Println(x)
9     }else{
10        cetak(x+1)
11        fmt.Println(x)
12    }
13 }
```

```
D:\DEV\DEMO>go build contoh.go
```

```
D:\DEV\DEMO>contoh.exe
```

```
10
9
8
7
6
5
```

Catatan:

- Teknik rekursif ini merupakan salah satu alternatif untuk mengganti struktur kontrol perulangan dengan memanfaatkan subprogram (bisa fungsi ataupun prosedur).
- Untuk menghentikan proses rekursif digunakan percabangan (if-then).
- Base-case adalah kondisi proses rekursif berhenti. Base-case merupakan hal terpenting dan pertama yang harus diketahui ketika akan membuat program rekursif. Mustahil membuat program rekursif tanpa mengetahui base-case terlebih dahulu.
- Recursive-case adalah kondisi dimana proses pemanggilan dirinya sendiri dilakukan. Kondisi recursive-case adalah komplemen atau negasi dari base-case.

- Setiap algoritma rekursif selalu memiliki padanan dalam bentuk algoritma iteratif.

II. GUIDED

Guided 1

Source Code

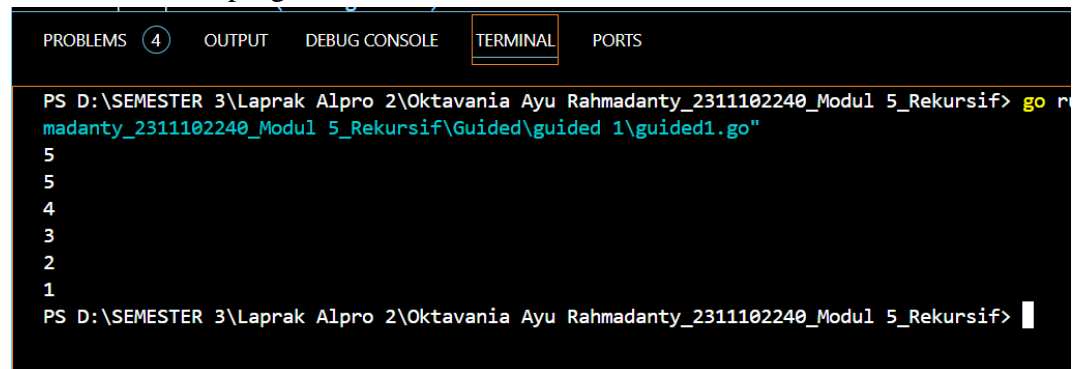
```
package main

import "fmt"

func main() {
    var n int
    fmt.Scan(&n) //membaca input pengguna
    baris(n)     //memanggil fungsi rekursif 'baris'
}

func baris(bilangan int) {
    if bilangan == 1 {
        fmt.Println(1)
    } else {
        fmt.Println(bilangan)
        baris(bilangan - 1)
    }
}
```

Screenshot hasil program



```
PS D:\SEMESTER 3\Laprak Alpro 2\Oktavania Ayu Rahmadanty_2311102240_Modul 5_Rekursif> go run madanty_2311102240_Modul 5_Rekursif\Guided\guided1\guided1.go
5
5
4
3
2
1
PS D:\SEMESTER 3\Laprak Alpro 2\Oktavania Ayu Rahmadanty_2311102240_Modul 5_Rekursif>
```

Penjelasan:

Program di atas adalah program Go yang membaca sebuah bilangan bulat n dari input pengguna dan kemudian mencetak bilangan tersebut secara menurun menggunakan fungsi rekursif bernama `baris`. Dalam fungsi `main`, program menerima input dari pengguna yang disimpan dalam variabel `n`, kemudian memanggil fungsi `baris` dengan parameter `n`. Fungsi `baris` akan mencetak nilai bilangan yang diterima sebagai argumen, lalu memanggil dirinya sendiri dengan nilai bilangan yang dikurangi satu. Rekursi ini akan

terus berlanjut hingga nilai bilangan mencapai 1, di mana fungsi mencetak angka 1 dan berhenti.

Guided 2

Source Code

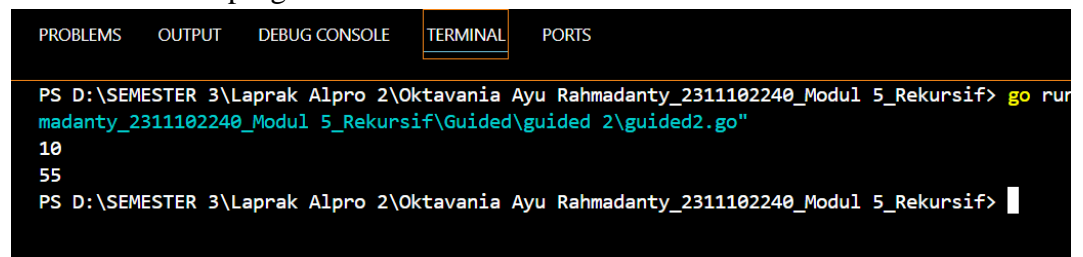
```
package main

import "fmt"

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

func penjumlahan(n int) int {
    if n == 1 {
        return 1
    } else {
        return n + penjumlahan(n-1)
    }
}
```

Screenshot hasil program



```
PS D:\SEMESTER 3\Laparak Alpro 2\Oktavania Ayu Rahmadanty_2311102240_Modul 5_Rekursif> go run madanty_2311102240_Modul 5_Rekursif\Guided\guided 2\guided2.go
10
55
PS D:\SEMESTER 3\Laparak Alpro 2\Oktavania Ayu Rahmadanty_2311102240_Modul 5_Rekursif>
```

Penjelasan:

Program di atas adalah program Go yang melakukan penjumlahan dari angka 1 hingga n menggunakan rekursi. Pertama, program membaca input bilangan bulat n dari pengguna dan mencetak hasil dari pemanggilan fungsi penjumlahan(n). Fungsi penjumlahan adalah fungsi rekursif yang menjumlahkan bilangan n dengan hasil penjumlahan bilangan-bilangan sebelumnya hingga mencapai 1. Jika nilai n adalah 1, fungsi

mengembalikan nilai 1, yang menjadi kondisi dasar untuk menghentikan rekursi. Jika tidak, fungsi menambahkan n dengan hasil rekursi dari penjumlahan($n-1$), sehingga total dari 1 hingga n dihitung dan dikembalikan.

III. UNGUIDED

Unguided 1

Source Code

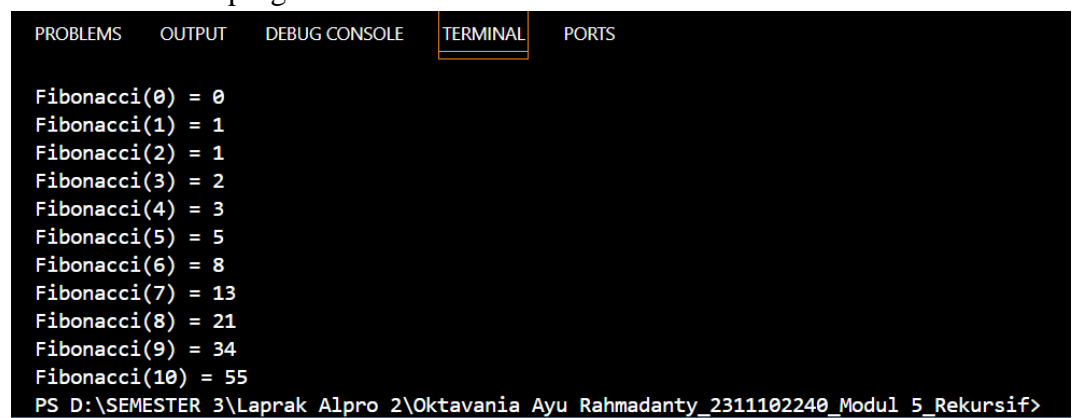
```
package main

import "fmt"

func fibonacci(n int) int {
    if n <= 1 {
        return n
    }
    return fibonacci(n-1) + fibonacci(n-2)
}

func main() {
    for i := 0; i <= 10; i++ {
        fmt.Printf("Fibonacci(%d) = %d\n", i, fibonacci(i))
    }
}
```

Screenshot hasil program

A screenshot of a terminal window showing the output of a Go program. The terminal has a black background with white text. At the top, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected and highlighted with a yellow border), and 'PORTS'. The output text lists the Fibonacci sequence from 0 to 10: 'Fibonacci(0) = 0', 'Fibonacci(1) = 1', 'Fibonacci(2) = 1', 'Fibonacci(3) = 2', 'Fibonacci(4) = 3', 'Fibonacci(5) = 5', 'Fibonacci(6) = 8', 'Fibonacci(7) = 13', 'Fibonacci(8) = 21', 'Fibonacci(9) = 34', and 'Fibonacci(10) = 55'. At the bottom of the terminal, the command prompt shows the file path: 'PS D:\SEMESTER 3\Laparak Alpro 2\Oktavania Ayu Rahmadanty_2311102240_Modul 5_Rekursif>'.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Fibonacci(0) = 0
Fibonacci(1) = 1
Fibonacci(2) = 1
Fibonacci(3) = 2
Fibonacci(4) = 3
Fibonacci(5) = 5
Fibonacci(6) = 8
Fibonacci(7) = 13
Fibonacci(8) = 21
Fibonacci(9) = 34
Fibonacci(10) = 55
PS D:\SEMESTER 3\Laparak Alpro 2\Oktavania Ayu Rahmadanty_2311102240_Modul 5_Rekursif>
```

Penjelasan:

Program di atas menggunakan bahasa Go (Golang) untuk menghitung deret Fibonacci secara rekursif. Fungsi `fibonacci(n)` mendefinisikan bahwa jika n bernilai 0 atau 1, maka ia akan mengembalikan 0 dan 1 secara langsung sebagai basis kasus, sedangkan untuk nilai n lebih dari 1, fungsi akan memanggil dirinya sendiri dengan `fibonacci(n-1) + fibonacci(n-2)` hingga mencapai basis kasus. Pada fungsi `main()`, program mencetak deret Fibonacci dari suku ke-0 hingga ke-10 menggunakan perulangan for. Hasil

akhirnya adalah deret Fibonacci yang sesuai dengan tabel pada soal, yaitu 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, dan 55.

Unguided 2

Source Code

```
package main

import "fmt"

func cetakBintang(n int) {
    if n == 0 {
        return
    }
    fmt.Print("*")
    cetakBintang(n - 1)
}

func polaBintang(n, current int) {
    if current > n {
        return
    }
    cetakBintang(current)
    fmt.Println()
    polaBintang(n, current+1)
}

func main() {
    var N int
    fmt.Print("Masukkan jumlah baris: ")
    fmt.Scan(&N)

    polaBintang(N, 1)
}
```

Screenshot hasil program

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

*****
PS D:\SEMESTER 3\Laprak Alpro 2\Oktavania Ayu Rahmadanty_2311102240_Modul 5_Rekursif> go run
madanty_2311102240_Modul 5_Rekursif\Unguided\Unguided 2\unguided2.go"
Masukkan jumlah baris: 1
*
PS D:\SEMESTER 3\Laprak Alpro 2\Oktavania Ayu Rahmadanty_2311102240_Modul 5_Rekursif> go run
madanty_2311102240_Modul 5_Rekursif\Unguided\Unguided 2\unguided2.go"
Masukkan jumlah baris: 3
*
**
***
PS D:\SEMESTER 3\Laprak Alpro 2\Oktavania Ayu Rahmadanty_2311102240_Modul 5_Rekursif> |
```

Penjelasan:

Program ini menggunakan dua fungsi rekursif untuk mencetak pola bintang bertingkat. Fungsi `cetakBintang` bertugas mencetak sejumlah bintang sesuai dengan parameter n, di mana setiap kali dipanggil, ia mencetak satu bintang dan mengurangi nilai n hingga n menjadi 0. Fungsi `polaBintang` mengatur jumlah baris bintang yang akan ditampilkan. Dimulai dari 1 bintang di baris pertama hingga mencapai N bintang di baris terakhir, fungsi ini memanggil dirinya sendiri dengan menambah nilai `current` (jumlah bintang yang akan dicetak) setiap kali setelah memanggil fungsi `cetakBintang`. Rekursi berhenti saat `current` melebihi nilai N yang diberikan pengguna.

Unguided 3

Source Code

```
package main

import "fmt"

func faktorRekursif(n, i int) {
    if i > n {
        return
    }
    if n%i == 0 {
        fmt.Print(i, " ")
    }
    faktorRekursif(n, i+1)
}
```

```
func main() {
    var N int
    fmt.Print("Masukkan bilangan: ")
    fmt.Scan(&N)

    fmt.Print("Faktor dari ", N, ": ")
    faktorRekursif(N, 1)
    fmt.Println()
}
```

Screenshot hasil program

```
PS D:\SEMESTER 3\Laprak Alpro 2\Oktavania Ayu Rahmadanty_2311102240_Modul 5_Rekursif> go run madanty_2311102240_Modul 5_Rekursif\Unguided\Unguided 3\unguided3.go
Masukkan bilangan: 5
Faktor dari 5: 1 5
PS D:\SEMESTER 3\Laprak Alpro 2\Oktavania Ayu Rahmadanty_2311102240_Modul 5_Rekursif> go run madanty_2311102240_Modul 5_Rekursif\Unguided\Unguided 3\unguided3.go
Masukkan bilangan: 12
Faktor dari 12: 1 2 3 4 6 12
PS D:\SEMESTER 3\Laprak Alpro 2\Oktavania Ayu Rahmadanty_2311102240_Modul 5_Rekursif> |
```

Penjelasan:

Program di atas menggunakan pendekatan rekursif untuk menampilkan faktor-faktor dari suatu bilangan positif N. Fungsi `faktorRekursif` menerima dua argumen: n (bilangan yang dicari faktornya) dan i (indeks mulai dari 1). Setiap kali fungsi dipanggil, ia memeriksa apakah bilangan n habis dibagi oleh i, jika iya, i dicetak sebagai faktor. Proses ini berlanjut dengan memanggil kembali fungsi tersebut dengan nilai i+1, hingga nilai i lebih besar dari n, yang menandakan akhir dari rekursi. Pada akhirnya, semua faktor yang habis membagi N akan dicetak dalam urutan yang benar.

Unguided 4

Source Code

```
package main

import (
    "fmt"
)
```

```

func printSequence(n, current int) {
    fmt.Printf("%d ", current)

    if current > 1 {
        printSequence(n, current-1)
    }

    if current < n {
        fmt.Printf("%d ", current)
    }
}

func main() {
    fmt.Print("Masukan 5: ")
    printSequence(5, 5)
    fmt.Println()

    fmt.Print("Masukan 9: ")
    printSequence(9, 9)
    fmt.Println()
}

```

Screenshot hasil program

The screenshot shows a terminal window with the following content:

```

PS D:\SEMESTER 3\LapraK Alpro 2\Oktavania Ayu Rahmadanty_2311102240_Modul 5_Rekursif> g
madanty_2311102240_Modul 5_Rekursif\Unguided\Unguided 4\unguided4.go"
Masukan 5: 5 4 3 2 1 1 2 3 4
Masukan 9: 9 8 7 6 5 4 3 2 1 1 2 3 4 5 6 7 8
PS D:\SEMESTER 3\LapraK Alpro 2\Oktavania Ayu Rahmadanty_2311102240_Modul 5_Rekursif>

```

Penjelasan:

Program di atas mencetak urutan angka yang menurun dari angka n ke 1, lalu kembali naik hingga mencapai n lagi, dengan menggunakan rekursi. Fungsi `printSequence` menerima dua parameter: n (batas atas angka) dan $current$ (angka yang saat ini sedang diproses). Pertama, angka $current$ dicetak, kemudian fungsi dipanggil lagi secara rekursif dengan nilai $current-1$ sampai mencapai 1. Setelah itu, jika nilai $current$ lebih kecil dari n , angka yang sama dicetak kembali untuk membuat urutan menaik. Ini menciptakan pola angka yang simetris, dimulai dari n , turun ke 1, dan naik kembali ke n .

Unguided 5

Source Code

```
package main

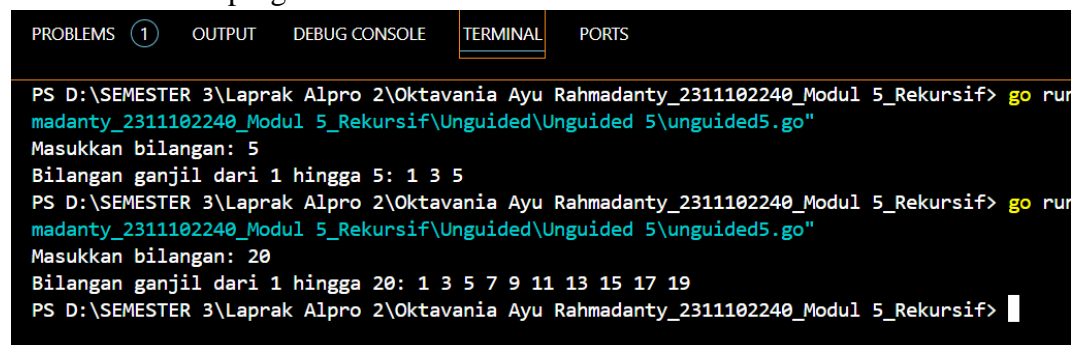
import "fmt"

func tampilkanGanjil(n, i int) {
    if i > n {
        return
    }
    if i%2 != 0 {
        fmt.Print(i, " ")
    }
    tampilkanGanjil(n, i+1)
}

func main() {
    var N int
    fmt.Print("Masukkan bilangan: ")
    fmt.Scan(&N)

    fmt.Print("Bilangan ganjil dari 1 hingga ", N, ": ")
    tampilkanGanjil(N, 1)
    fmt.Println()
}
```

Screenshot hasil program



```
PS D:\SEMESTER 3\Laprak Alpro 2\Oktavania Ayu Rahmadanty_2311102240_Modul 5_Rekursif> go run madanty_2311102240_Modul 5_Rekursif\Unguided\Unguided 5\unguided5.go
Masukkan bilangan: 5
Bilangan ganjil dari 1 hingga 5: 1 3 5
PS D:\SEMESTER 3\Laprak Alpro 2\Oktavania Ayu Rahmadanty_2311102240_Modul 5_Rekursif> go run madanty_2311102240_Modul 5_Rekursif\Unguided\Unguided 5\unguided5.go
Masukkan bilangan: 20
Bilangan ganjil dari 1 hingga 20: 1 3 5 7 9 11 13 15 17 19
PS D:\SEMESTER 3\Laprak Alpro 2\Oktavania Ayu Rahmadanty_2311102240_Modul 5_Rekursif>
```

Penjelasan:

Program ini menggunakan rekursi untuk menampilkan deretan bilangan ganjil dari 1 hingga N, di mana N adalah bilangan bulat positif yang dimasukkan oleh pengguna. Fungsi `tampilkanGanjil` menerima dua parameter: `n` (batas atas) dan `i` (indeks awal yang dimulai dari 1). Pada setiap

panggilan rekursif, fungsi memeriksa apakah i adalah bilangan ganjil (dengan menggunakan `i % 2 != 0`), lalu mencetaknya jika benar. Fungsi ini kemudian memanggil dirinya sendiri dengan nilai $i+1$ sampai semua bilangan ganjil dari 1 hingga N telah ditampilkan, dengan rekursi berhenti saat i melebihi N .

Unguided 6

Source Code

```
package main

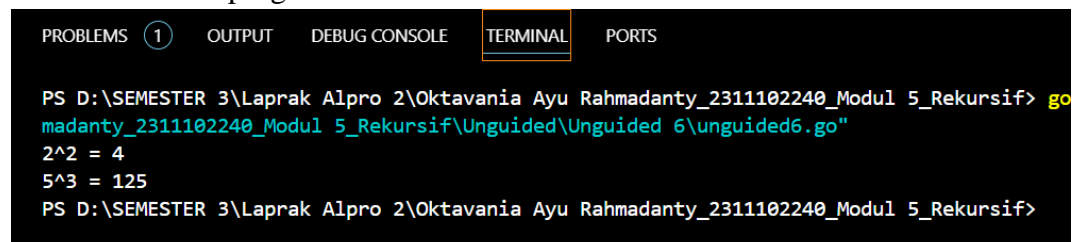
import (
    "fmt"
)

func power(x, y int) int {
    if y == 0 {
        return 1 // Basis kasus:  $x^0 = 1$ 
    }
    return x * power(x, y-1) // Rekursif:  $x * (x^{(y-1)})$ 
}

func main() {
    // Input contoh dari soal
    var x1, y1 = 2, 2
    var x2, y2 = 5, 3

    fmt.Printf("%d^%d = %d\n", x1, y1, power(x1, y1))
    fmt.Printf("%d^%d = %d\n", x2, y2, power(x2, y2))
}
```

Screenshot hasil program



```
PS D:\SEMESTER 3\Laprak Alpro 2\Oktavania Ayu Rahmadanty_2311102240_Modul 5_Rekursif> go run madanty_2311102240_Modul 5_Rekursif\Unguided\Unguided 6\unguided6.go
2^2 = 4
5^3 = 125
PS D:\SEMESTER 3\Laprak Alpro 2\Oktavania Ayu Rahmadanty_2311102240_Modul 5_Rekursif>
```

Penjelasan:

Program di atas menghitung hasil perpangkatan (x^y) menggunakan fungsi rekursif dalam bahasa Go. Fungsi `power(x, y)` memiliki basis kasus

saat $y == 0$, yang mengembalikan 1 karena $(x^0 = 1)$. Jika $y > 0$, fungsi akan memanggil dirinya sendiri dengan mengembalikan hasil perkalian $x * \text{power}(x, y-1)$ hingga mencapai basis kasus. Pada fungsi `main()`, program menerima dua contoh input, yaitu (2^2) dan (5^3) , lalu mencetak hasilnya, yaitu 4 dan 125, sesuai dengan keluaran yang diharapkan. Program ini hanya menggunakan operator `*` tanpa library `math`, sesuai instruksi soal.