

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN DUA**

**MODUL VI**

**REKURSIF**



Oleh:

YAYANG ALYA BILQIS

2311102229

IF 11 02

**S1 TEKNIK INFORMATIKA  
TELKOM UNIVERSITY PURWOKERTO**

**2024**

## I. DASAR TEORI

- Pengantar Rekursif

**Rekursif** merupakan cara menyelesaikan suatu masalah dengan menyelesaikan *sub-masalah* terlebih dahulu yang memiliki keterkaitan dengan masalah utama. Pada rekursif biasanya ditambahkan struktur control percabangan (*if-then*) untuk menghentikan proses rekursif, kondisinya disebut dengan *base-case* dimana pada yang bernilai true proses rekursif akan berhenti. Pada rekursif, ada yang dinamakan *recursive-case* yaitu *kasus pemanggilan diri sendiri dimana recursive-case merupakan negasi dari kondisi base-case atau ketika nilai  $x \neq 10$* . Sebagai contoh:

	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	

```
package main
import "fmt"
func main(){
    cetak(5)
}
func cetak(x int){
    fmt.Println(x)
    cetak(x+1)
}
```

```
procedure cetak(in x:integer)
algoritma
    if x == 10 then
        output(x)
    else
        output(x)
        cetak(x+1)
    endif
endprocedure
```

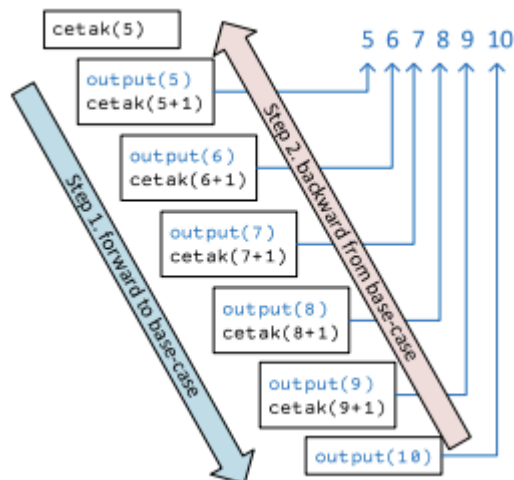
```

package main
import "fmt"
func main(){
    cetak(5)
}
func cetak(x int){
    if x == 10 {
        fmt.Println(x)
    }else{
        fmt.Println(x)
        cetak(x+1)
    }
}

```

Program ini merupakan contoh dari fungsi rekursif, dimana fungsi 'cetak' akan memanggil dirinya sendiri dengan nilai x, dalam setiap panggilan maka akan bertambah 1. Rekursif

akan terjadi apabila fungsi memanggil dirinya sendiri, baik secara langsung maupun tidak langsung. Program ini akan mencetak nilai x yang terus meningkat tanpa batas karena tidak ada kondisi base-case, maka hasilnya rekursi itu tak terbatas dan penyebab nya bisa terjadi stack overflow.



Pada contoh kedua apabila subprogram dipanggil secara rekursif, maka subprogram akan terus menerus melakukan pemanggilan (forward) alhasil terjadi pemberhentian pada saat kondisi base-case nya sudah terpenuhi atau true. Proses yang akan terjadi selanjutnya adalah backward atau akan dikembalikan ke subprogram sebelumnya. Setelah semua intruksi cetak(10) selesai dieksekusi, maka program akan kembali ke cetak(9) yang

tadinya memanggil cetak (10). Proses akan terus terjadi hingga kembali ke cetak(5)

```
package main
import "fmt"
func main(){
    cetak(5)
}
func cetak(x int){
    if x == 10 {
        fmt.Println(x)
    }else{
        cetak(x+1)
        fmt.Println(x)
    }
}
```

Contoh tersebut merupakan program yang menghitung factorial dari sebuah bilangan dimana base-case akan berhenti memanggil dirinya sendiri dan fungsi mengembalikan nilai 1, base-case sangat penting untuk mencegah terjadinya rekursi tak terbatas dan fungsi bisa berhenti. Dalam contoh tersebut program akan menghitung 5! Dimana factorial akan memanggil dirinya sendiri dengan parameter n-1 setiap kali fungsi diulang hingga mencapai base case maka n akan dikurail 1

Fungsi merupakan satu kesatuan rangkaian intruksi yang memberikan atau menghasilkan suatu nilai dan biasanya memetakan input ke satu nilai yang lain. Fungsi akan selalu bekerja dengan cara menghasilkan/mengembalikan nilai. Fungsi mencakup adanya deklarasi tipe nilai yang dikembalikan, dan terdapat kata kunci return dalam sub program. Fungsi digunakan untuk assignment nilai ke suatu variable, bagian dari ekspresi, argumen, dan suatu sub program.

- **Komponen Rekursif**

Rekursif memiliki beberapa komponen kunci yang memungkinkan sebuah fungsi memanggil dirinya sendiri untuk menyelesaikan sub-masalah.

- Base case  
Kondisi ini menentukan kapan fungsi rekursif harus berhenti memanggil dirinya sendiri. Tanpa base-case, fungsi akan memanggil dirinya sendiri tanpa henti sehingga akan terjadi stack overflow yaitu pemanggilan tanpa batas. Base case merupakan case yang paling sederhana dari masalah yang sedang dalam pemecahan.
- Recursive case  
Merupakan bagian dari fungsi rekursif yang akan memanggil dirinya sendiri untuk memecah masalah menjadi sub yang lebih kecil

sehingga mudah dikelola dan mencapai kondisi base yang akan menghentikan rekursif

Deklarasi fungsi merupakan proses dimana sebuah fungsi akan didefinisikan, termasuk nama fungsi, parameter dan tipe data yang akan dikembalikan. Deklarasi fungsi adalah elemen fundamental dalam pemrograman yang memungkinkan kita untuk menulis kode yang lebih terstruktur.

Berikut adalah contoh fungsi untuk menghitung volume dari tabung apabila jari-jari alas dan tinggi tabung diketahui.

	Notasi Algoritma
1	function volumeTabung(jari_jari,tinggi : integer) -> real
2	kamus
3	luasAlas, volume: real
4	algoritma
5	luasAlas <- 3.14 * (jari_jari * jari_jari)
6	volume <- luasAlas * tinggi
7	return volume
8	endfunction
	Notasi dalam bahasa Go
10	func volumeTabung(jari_jari,tinggi int) float64 {
11	var luasAlas,volume float64
12	luasAlas = 3.14 * float64(jari_jari * jari_jari)
13	volume = luasAlas * tinggi
14	return volume
15	}

- **Cara Pemanggilan Fungsi**

Pemanggilan fungsi dilakukan dengan penulisan nama fungsi beserta argumen yang diminta oleh parameter dari fungsi. Pemanggilan fungsi bisa di assign ke suatu variable, menjadi bagian dari ekspresi, dan argument dari suatu subprogram. Fungsi bisa di assign ke suatu variable pada saat pemanggilan, bisa dioperasikan sesuai dengan tipe data yang dikembalikan, dan juga bisa langsung ditampilkan dengan output ataupun print.

## II. GUIDED

### Guided 1

```
package main

import "fmt"
```

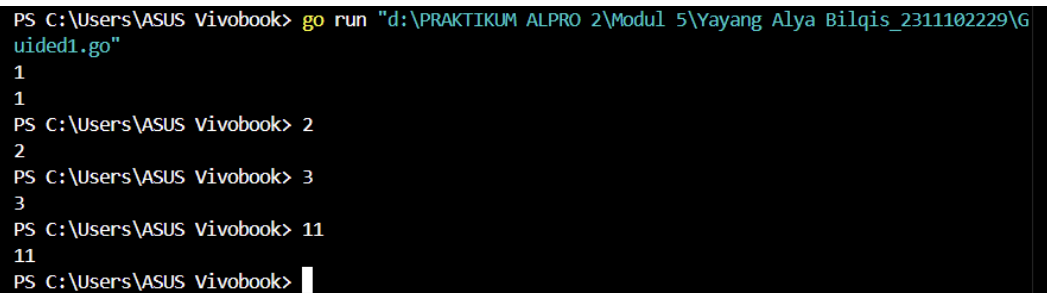
```

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

func baris(bilangan int) {
    if bilangan == 1 { // Base case: Jika bilangan sama
dengan 1
        fmt.Println(1) // Cetak angka 1
    } else { // Jika bilangan lebih besar dari 1
        fmt.Println(bilangan) // cetak bilangan saat ini
        baris(bilangan - 1)    // Panggil fungsi 'baris'
dengan bilangan
    }
}
}

```

### Screenshot Program



```

PS C:\Users\ASUS Vivobook> go run "d:\PRAKTIKUM ALPRO 2\Modul 5\Yayang Alya Bilqis_2311102229\Guided1.go"
1
1
PS C:\Users\ASUS Vivobook> 2
2
PS C:\Users\ASUS Vivobook> 3
3
PS C:\Users\ASUS Vivobook> 11
11
PS C:\Users\ASUS Vivobook> 

```

### Deskripsi Program

Program diatas adalah untuk fungsi akan memanggil dirinya sendiri dengan nilai yang lebih kecil sehingga kembali ke kondisi awal. Apabila rekursi terlalu dalam, maka akan menyebabkan stack overflow.

## Guided 2

```
package main

import "fmt"

// Fungsi buat menghitung faktorial
func factorial(n int) int {
    if n == 0 || n == 1 {
        return 1
    }
    result := 1
    for i := 2; i <= n; i++ {
        result *= i
    }
    return result
}

// Fungsi buat menghitung permutasi
func permutation(n, r int) int {
    return factorial(n) / factorial(n-r)
}

// Fungsi buat menghitung kombinasi
func combination(n, r int) int {
    return factorial(n) / (factorial(r) * factorial(n-r))
}

func main() {
    // Input 4 bilangan
    var a, b, c, d int
```

```

        fmt.Println("Masukkan bilangan a, b, c, d (dengan
spasi): ")

        fmt.Scanf("%d %d %d %d", &a, &b, &c, &d)

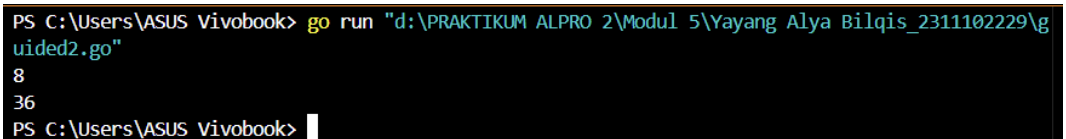
        // Cek syarat a >= c dan b >= d
        if a >= c && b >= d {
            // Menghitung permutasi dan kombinasi a dan c
            permutasiAC := permutation(a, c)
            kombinasiAC := combination(a, c)

            // Menghitung permutasi dan kombinasi b dan d
            permutasiBD := permutation(b, d)
            kombinasiBD := combination(b, d)

            // Output hasil
            fmt.Println("Permutasi(a, c) dan Kombinasi(a,
c):", permutasiAC, kombinasiAC)
            fmt.Println("Permutasi(b, d) dan Kombinasi(b,
d):", permutasiBD, kombinasiBD)
        } else {
            fmt.Println("Syarat a >= c dan b >= d tidak
terpenuhi.")
        }
    }
}

```

## Screenshot Program



```

PS C:\Users\ASUS Vivobook> go run "d:\PRAKTIKUM ALPRO 2\Modul 5\Yayang Alya Bilqis_2311102229\g
uided2.go"
8
36
PS C:\Users\ASUS Vivobook>

```

## Deskripsi Program



Program ini berfungsi untuk menghitung bilangan bulat dengan batas dimana inputan yang akan diterima memanggil fungsi penjumlahan dengan nilai n. Jika nilai n adalah 1, maka fungsi akan mengembalikan nilai 1. Program ini akan berulang terus menerus sehingga mencapai kondisi dasar (nilai n menjadi 1)

### III. UNGUIDED

#### UNGUIDED 1

```
package main

import "fmt"

var memo = map[int]int{0: 0, 1: 1}

func fibonacci(n int) int {
    if val, ok := memo[n]; ok {
        return val
    }
    if n <= 1 {
        return n
    }
    result := fibonacci(n-1) + fibonacci(n-2)
    memo[n] = result
    return result
}

func fibonacciIterative(n int) int {
    if n <= 1 {
        return n
    }
    prev, current := 0, 1
```

```

        for i := 2; i <= n; i++ {
            next := prev + current
            prev, current = current, next
        }
        return current
    }

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

        fmt.Println("\nMenggunakan iterasi:")
        for i := 0; i <= 10; i++ {
            fmt.Printf("Fibonacci(%d) = %d\n", i,
                fibonacciIterative(i))
        }
    }
}

```

## Screenshot Program

```

PS C:\Users\ASUS Vivobook> go run "d:\PRAKTIKUM ALPRO 2\Modul 5\Yayang Alya Bilqis_2311102229\un
guided1.go"
Menggunakan rekursi dengan memoization:
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

```

```
Menggunakan iterasi:
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 C:\Users\ASUS_Vivobook>
```

## Deskripsi Program

Program ini akan merekursi dengan menyimpan hasil perhitungan yang telah dihitung sebelumnya dan menggunakan iterasi untuk menghitung deret secara langsung lalu fungsi akan membandingkan hasil performanya.

## UNGUIDED 2

```
package main

import (
    "fmt"
    "os"
    "strconv"
    "strings"
)

func cetakBintang(n int, simbol rune, ascending bool) {
    if ascending {
        if n > 0 {
            cetakBintang(n-1, simbol, ascending)
            fmt.Println(string(ulangi(simbol, n)))
        }
    } else {
        if n > 0 {
            fmt.Println(string(ulangi(simbol, n)))
        }
    }
}
```

```

        cetakBintang(n-1, simbol, ascending)

    }

}

func ulangi(karakter rune, jumlah int) []rune {
    var output []rune
    for i := 0; i < jumlah; i++ {
        output = append(output, karakter)
    }
    return output
}

func validasiInput(input string) (int, error) {
    jumlah, err := strconv.Atoi(input)
    if err != nil || jumlah < 1 {
        return 0, fmt.Errorf("input tidak valid")
    }
    return jumlah, nil
}

func pilihSimbol() rune {
    var simbol string

    fmt.Print("Masukkan simbol yang ingin digunakan\n(default: *): ")

    fmt.Scanln(&simbol)

    if simbol == "" {
        return '*'
    }

    return rune(simbol[0])
}

```

```
func main() {  
    var pilihan string  
    fmt.Println("Pilih mode pola:")  
    fmt.Println("1. Pola Meningkat")  
    fmt.Println("2. Pola Menurun")  
    fmt.Print("Pilihan: ")  
    fmt.Scanln(&pilihan)  
  
    fmt.Print("Masukkan jumlah baris: ")  
    var input string  
    fmt.Scanln(&input)  
    baris, err := validasiInput(input)  
    if err != nil {  
        fmt.Println(err)  
        os.Exit(1)  
    }  
  
    simbol := pilihSimbol()  
  
    switch strings.TrimSpace(pilihan) {  
    case "1":  
        fmt.Println("Pola Meningkat:")  
        cetakBintang(baris, simbol, true)  
    case "2":  
        fmt.Println("Pola Menurun:")  
        cetakBintang(baris, simbol, false)  
    default:  
        fmt.Println("Pilihan tidak valid.")  
        os.Exit(1)  
    }  
}
```

## Screenshot Program

```
PS C:\Users\ASUS Vivobook> go run "d:\PRAKTIKUM ALPRO 2\Modul 5\Yayang Alya Bilqis_2311102229\unguided2.go"
Pilih mode pola:
1. Pola Meningkat
2. Pola Menurun
Pilihan: 1
Masukkan jumlah baris: 5
Masukkan simbol yang ingin digunakan (default: *): *
Pola Meningkat:
*
**
***
****
*****
PS C:\Users\ASUS Vivobook>
```

```
Pilihan: 2
Masukkan jumlah baris: 9
Masukkan simbol yang ingin digunakan (default: *): *
Pola Menurun:
*****
*****
*****
*****
*****
****
***
**
*
PS C:\Users\ASUS Vivobook>
```

## Deskripsi Program

Program ini akan mencetak Bintang dalam pola meningkat atau menurun yang diminta oleh user, sebelum mencetak Bintang maka fungsi akan memanggil dirinya sendiri dengan n-1. Jika true maka fungsi akan mencetak Bintang terlebih dahulu sebelum dirinya sendiri.

## UNGUIDED 3

```
package main

import (
    "fmt"
    "os"
    "strconv"
)
```

```
func faktorRekursif(n, divisor int) {  
    if divisor > n {  
        return  
    }  
    if n%divisor == 0 {  
        fmt.Print(divisor, " ")  
    }  
    faktorRekursif(n, divisor+1)  
}  
  
func validasiInput(input string) (int, error) {  
    jumlah, err := strconv.Atoi(input)  
    if err != nil || jumlah < 1 {  
        return 0, fmt.Errorf("input tidak valid")  
    }  
    return jumlah, nil  
}  
  
func main() {  
    fmt.Print("Masukkan bilangan bulat positif: ")  
    var input string  
    fmt.Scanln(&input)  
    n, err := validasiInput(input)  
    if err != nil {  
        fmt.Println(err)  
        os.Exit(1)  
    }  
  
    fmt.Printf("Faktor dari %d adalah: ", n)  
    faktorRekursif(n, 1)
```

```
        fmt.Println()

    }

    func pilihSimbol() rune {
        var simbol string

        fmt.Print("Masukkan simbol yang ingin digunakan
(default: *): ")

        fmt.Scanln(&simbol)

        if simbol == "" {
            return '*'
        }

        return rune(simbol[0])
    }

    func main() {
        var pilihan string

        fmt.Println("Pilih mode pola:")
        fmt.Println("1. Pola Meningkat")
        fmt.Println("2. Pola Menurun")
        fmt.Print("Pilihan: ")
        fmt.Scanln(&pilihan)

        fmt.Print("Masukkan jumlah baris: ")
        var input string
        fmt.Scanln(&input)
        baris, err := validasiInput(input)
        if err != nil {
            fmt.Println(err)
            os.Exit(1)
        }

        simbol := pilihSimbol()
```

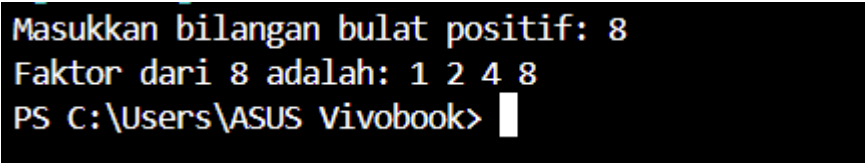


```

        switch strings.TrimSpace(pilihan) {
        case "1":
            fmt.Println("Pola Meningkat:")
            cetakBintang(baris, simbol, true)
        case "2":
            fmt.Println("Pola Menurun:")
            cetakBintang(baris, simbol, false)
        default:
            fmt.Println("Pilihan tidak valid.")
            os.Exit(1)
        }
    }
}

```

### Screenshot Program



```

Masukkan bilangan bulat positif: 8
Faktor dari 8 adalah: 1 2 4 8
PS C:\Users\ASUS Vivobook>

```

### Deskripsi Program

Program ini akan mencari factor dari bilangan n dan akan meningkat setiap kali fungsi dipanggil kembali. Fungsi akan memvalidasi inputan user untuk memastikan inputan merupakan bilangan bulat positif. Jika inputan tidak valid, maka fungsi akan merunning salah.

### UNGUIDED 4

```

package main

import (
    "fmt"
    "os"
    "strconv"

```

```

    )

    // Fungsi rekursif untuk mencetak deret angka dari N hingga 1
    dan kembali ke N
    func cetakDeret(n, current int) {
        if current > n {
            return
        }
        fmt.Print(current, " ")
        if current == 1 {
            for i := 2; i <= n; i++ {
                fmt.Print(i, " ")
            }
            return
        }
        cetakDeret(n, current-1)
        fmt.Print(current, " ")
    }

    // Fungsi validasi input untuk memastikan input adalah
    bilangan bulat positif
    func validasiInput(input string) (int, error) {
        angka, err := strconv.Atoi(input)
        if err != nil || angka < 1 {
            return 0, fmt.Errorf("input tidak valid, harap
            masukkan bilangan bulat positif")
        }
        return angka, nil
    }

    func main() {
        fmt.Print("Masukkan bilangan bulat positif N: ")
    }

```

```

        var input string

        fmt.Scanln(&input)

        n, err := validasiInput(input)

        if err != nil {

            fmt.Println(err)

            os.Exit(1)

        }

        fmt.Printf("Deret dari %d hingga 1 dan kembali ke %d:
", n, n)

        cetakDeret(n, n)

        fmt.Println()

    }

```

## Screenshot Program

```

PS C:\Users\ASUS Vivobook> go run "d:\PRAKTIKUM ALPRO 2\Modul 5\Yayang Alya Bilqis_2311102229\un
guided4.go"
Masukkan bilangan bulat positif N: 10
Deret dari 10 hingga 1 dan kembali ke 10: 10 9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9 10 2 3 4 5 6 7 8
9 10
PS C:\Users\ASUS Vivobook>

```

## Deskripsi Program

Program ini akan menerima dua parameter dan fungsi akan mencetak angka dari current hingga 1 dan kembali n. Jika current sama dengan 1, fungsi akan mencetak angka dari 1 dan kemudian kembali mencetak dari 2 hingga n. fungsi akan memastikan bahwa inputan dari user merupakan bilangan positif. Program ini menunjukkan penggunaan rekursi dengan mencetak deret dari n hingga ke 1 kemudian kembali ke n.

## UNGUIDED 5

```

package main

import (

```

```
    "fmt"
    "os"
    "strconv"
)

func cetakBilanganGanjil(n, current int) {
    if current > n {
        return
    }
    fmt.Print(current, " ")
    cetakBilanganGanjil(n, current+2)
}

func validasiInput(input string) (int, error) {
    angka, err := strconv.Atoi(input)
    if err != nil || angka < 1 {
        return 0, fmt.Errorf("Input tidak valid, harap masukkan bilangan bulat positif")
    }
    return angka, nil
}

func main() {
    fmt.Print("Masukkan bilangan bulat positif N: ")
    var input string
    fmt.Scanln(&input)
    n, err := validasiInput(input)
    if err != nil {
        fmt.Println(err)
        os.Exit(1)
    }
}
```

```

        fmt.Printf("Deret bilangan ganjil dari 1 hingga %d
adalah: ", n)

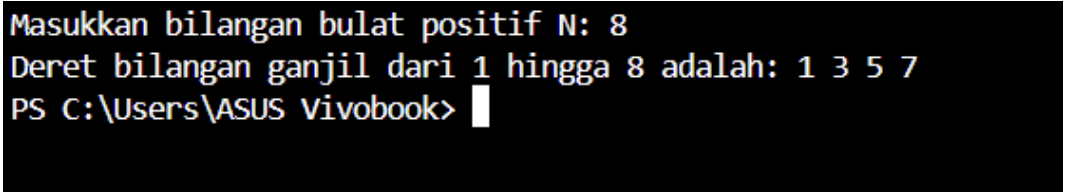
        cetakBilanganGanjil(n, 1)

        fmt.Println()

    }

```

## Screenshot Program



```

Masukkan bilangan bulat positif N: 8
Deret bilangan ganjil dari 1 hingga 8 adalah: 1 3 5 7
PS C:\Users\ASUS Vivobook>

```

## Deskripsi Program

Program ini akan menerima dua parameter dan fungsi akan mencetak angka dari current. Program menunjukkan penggunaan rekursi untuk mencetak deret bilangan ganjil dari 1 hingga n. fungsi akan memvalidasi input dari user untuk memastikan apakah bilangannya positif atau bukan.

## UNGUIDED 6

```

package main

import (
    "fmt"
)

func power(base, exponent int) int {
    if exponent == 0 {
        return 1
    } else if exponent < 0 {
        return 1 / power(base, -exponent)
    } else {

```

```

        return base * power(base, exponent-1)

    }

}

func powerIterative(base, exponent int) int {
    result := 1
    for i := 0; i < exponent; i++ {
        result *= base
    }
    return result
}

func printResult(base, exponent, result int) {
    fmt.Printf("%d pangkat %d = %d\n", base, exponent,
result)
}

func main() {
    var base, exponent int

    fmt.Print("Masukkan bilangan pokok: ")
    fmt.Scan(&base)

    fmt.Print("Masukkan pangkat: ")
    fmt.Scan(&exponent)

    recursiveResult := power(base, exponent)

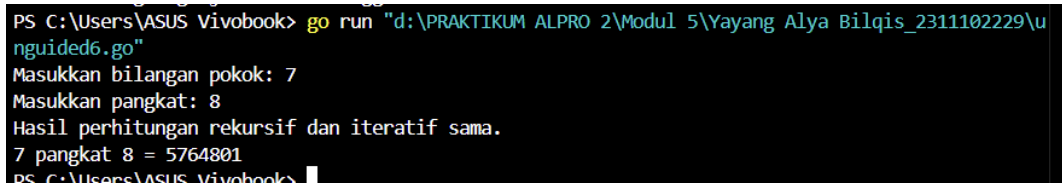
    iterativeResult := powerIterative(base, exponent)

    if recursiveResult == iterativeResult {

```

```
        fmt.Println("Hasil perhitungan rekursif dan  
        iteratif sama.")  
    } else {  
        fmt.Println("Terdapat kesalahan dalam  
        perhitungan.")  
    }  
  
    printResult(base, exponent, recursiveResult)  
}
```

### Screenshot Program



```
PS C:\Users\ASUS Vivobook> go run "d:\PRAKTIKUM ALPRO 2\Modul 5\Yayang Alya Bilqis_2311102229\un-  
guided6.go"  
Masukkan bilangan pokok: 7  
Masukkan pangkat: 8  
Hasil perhitungan rekursif dan iteratif sama.  
7 pangkat 8 = 5764801  
PS C:\Users\ASUS Vivobook>
```

### Deskripsi Program

Program ini akan mengembalikan basis eksponen negative menjadi positif dan akan melakukan perulangan untuk menghitung pangkat secara iterative. Program ini akan meminta user untuk memasukkan bilangan pokok dan kemudian memanggil kedua fungsi untuk membandingkan hasil.