

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2
MODUL V
REKURSIF**



Disusun Oleh :

Erwin Rivaldo Silaban

IF-11-06

Dosen Pengampu :

Abednego Dwi Septiadi, S.Kom., M.Kom

**PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2024**

I. DASAR TEORI

Rekursif adalah proses atau prosedur di mana suatu fungsi memanggil dirinya sendiri berulang kali. Karena proses dalam rekursif ini berlangsung secara berulang, perlu adanya kondisi yang membatasi pengulangan tersebut. Jika tidak, proses tersebut tidak akan berhenti hingga memori yang digunakan untuk menampung proses itu penuh dan tidak lagi mampu menampungnya.

Kelebihan fungsi rekursif adalah program menjadi lebih singkat. Dalam beberapa kasus, penggunaan fungsi rekursif lebih mudah, seperti pada perhitungan pangkat, faktorial, deret Fibonacci, dan beberapa proses deret lainnya. Fungsi rekursif juga bisa lebih efisien dan cepat dibandingkan dengan proses iteratif. Namun, kekurangan fungsi rekursif adalah penggunaan memori yang lebih besar, karena setiap kali fungsi tersebut dipanggil, diperlukan sejumlah ruang memori tambahan. Rekursif juga berisiko tidak berhenti sehingga memori bisa habis terpakai, menyebabkan program menjadi hang.

II. GUIDED

1. Soal 1

- Source code

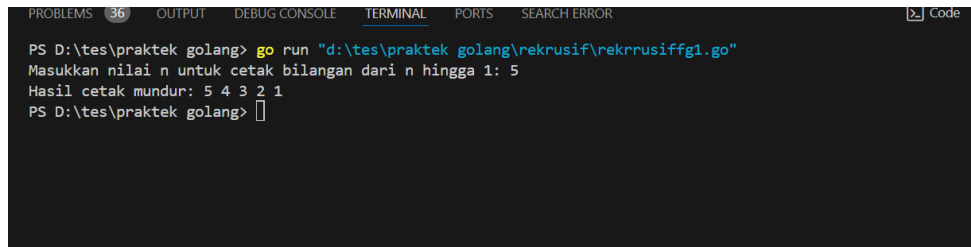
```
package main

import "fmt"

func cetakMundur(n int) {
    if n == 1 {
        fmt.Println(n)
        return
    }
    fmt.Print(n, " ")
    cetakMundur(n - 1)
}

func main() {
    var n int
    fmt.Print("Masukkan nilai n untuk cetak bilangan dari n hingga 1: ")
    fmt.Scanln(&n)
    fmt.Print("Hasil cetak mundur: ")
    cetakMundur(n)
}
```

- Screenshot hasil



```
PROBLEMS 36 OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR Code
PS D:\tes\praktek golang> go run "d:\tes\praktek golang\rekursif\rekrrusifg1.go"
Masukkan nilai n untuk cetak bilangan dari n hingga 1: 5
Hasil cetak mundur: 5 4 3 2 1
PS D:\tes\praktek golang>
```

- Deskripsi Program

Program ini mencetak bilangan secara mundur dari nilai `n` hingga 1 menggunakan fungsi rekursif. Fungsi `cetakMundur` menerima parameter `n` yang merupakan bilangan bulat. Dalam fungsi ini, terdapat kondisi dasar: jika `n` bernilai 1, program akan mencetak angka 1 dan berhenti (menggunakan `return`). Jika `n` lebih dari 1, program akan mencetak nilai `n` diikuti oleh spasi, lalu memanggil dirinya sendiri dengan nilai `n-1`, sehingga mencetak angka berikutnya hingga mencapai 1. Fungsi `main` meminta pengguna memasukkan nilai `n`, lalu memanggil fungsi `cetakMundur` untuk memulai pencetakan mundur dari `n` hingga 1.

2. Soal 2

- Source code

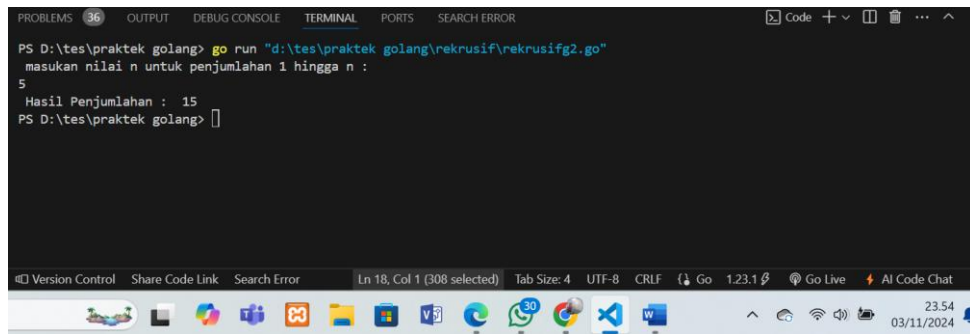
```
package main

import "fmt"

func jumlahRekrusifn(n int) int {
    if n == 1 {
        return 1
    }
    return n + jumlahRekrusifn(n-1)
}

func main() {
    var n int
    fmt.Println(" masukan nilai n untuk penjumlahan 1 hingga n : ")
    fmt.Scanln(&n)
    fmt.Println(" Hasil Penjumlahan : ", jumlahRekrusifn(n))
}
```

- Screenshot hasil



```
PS D:\tes\praktek golang> go run "d:\tes\praktek golang\rekursif\rekusifg2.go"
masukan nilai n untuk penjumlahan 1 hingga n :
5
Hasil Penjumlahan : 15
PS D:\tes\praktek golang>
```

- Deskripsi program

Program ini menghitung jumlah bilangan dari 1 hingga `n` menggunakan fungsi rekursif. Fungsi `jumlahRekrusifn` menerima parameter `n` dan memiliki kondisi dasar: jika `n` bernilai 1, fungsi mengembalikan nilai 1. Jika `n` lebih dari 1, fungsi akan menambahkan nilai `n` dengan hasil dari pemanggilan dirinya sendiri (`jumlahRekrusifn(n-1)`), sehingga menghasilkan penjumlahan berurutan dari `n` hingga 1. Fungsi `main` meminta pengguna memasukkan nilai `n`, lalu memanggil `jumlahRekrusifn` untuk menghitung dan mencetak hasil penjumlahan dari 1 hingga `n`.

3. Soal 3

- Source code

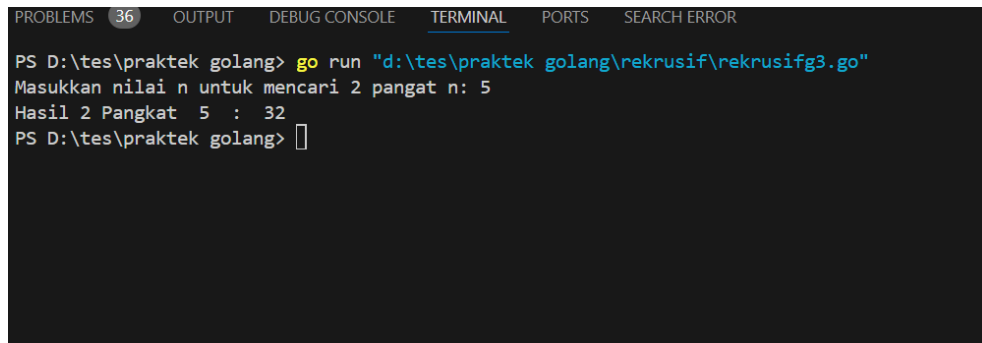
```
package main

import "fmt"

func pangkatDua(n int) int {
    if n == 0 {
        return 1
    }
    return 2 * pangkatDua(n-1)
}

func main() {
    var n int
    fmt.Print("Masukkan nilai n untuk mencari 2 pangkat n: ")
    fmt.Scanln(&n)
    fmt.Println("Hasil 2 Pangkat ", n, " : ", pangkatDua(n))
}
```

- Screenashot hasil



```
PROBLEMS 36 OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
PS D:\tes\praktek golang> go run "d:\tes\praktek golang\rekrusif\rekrusifg3.go"
Masukkan nilai n untuk mencari 2 pangkat n: 5
Hasil 2 Pangkat 5 : 32
PS D:\tes\praktek golang> 
```

- Deskripsi program

Program ini menghitung nilai 2 pangkat `n` menggunakan fungsi rekursif. Fungsi `pangkatDua` menerima parameter `n` dan memiliki kondisi dasar: jika `n` bernilai 0, fungsi mengembalikan 1 (karena 2 pangkat 0 adalah 1). Jika `n` lebih dari 0, fungsi akan mengalikan 2 dengan hasil pemanggilan dirinya sendiri (`pangkatDua(n-1)`), sehingga menghitung 2 pangkat `n`. Fungsi `main` meminta pengguna memasukkan nilai `n`, lalu memanggil `pangkatDua` untuk menghitung dan mencetak hasil 2 pangkat `n`.

4. Soal 4

- Source code

```
package main

import "fmt"

func pangkatDua(n int) int {
    if n == 0 {
        return 1
    }
    return 2 * pangkatDua(n-1)
}

func main() {
    var n int
    fmt.Print("Masukkan nilai n untuk mencari 2 pangkat n: ")
    fmt.Scanln(&n)
    fmt.Println("Hasil 2 Pangkat ", n, " : ", pangkatDua(n))
}
```

- Screenshot program

```
PS D:\tes\praktek golang> go run "d:\tes\praktek golang\rekursif\rekursifg4.go"
Masukkan nilai n untuk mencari 2 pangkat n: 7
Hasil 2 Pangkat 7 : 128
PS D:\tes\praktek golang>
```

- Deskripsi program

Program ini menghitung nilai 2 pangkat `n` secara rekursif. Fungsi `pangkatDua` menerima parameter `n` dan memiliki kondisi dasar: jika `n` bernilai 0, fungsi mengembalikan 1 (karena 2 pangkat 0 adalah 1). Jika `n` lebih dari 0, fungsi akan mengalikan 2 dengan hasil pemanggilan dirinya sendiri (`pangkatDua(n-1)`), sehingga menghitung 2 pangkat `n`. Di dalam fungsi `main`, pengguna diminta untuk memasukkan nilai `n`, lalu program memanggil `pangkatDua` untuk menghitung dan menampilkan hasilnya, yaitu nilai 2 pangkat `n`.

III. UNGUIDED

1. Unguided 1

Studi case

XXXXXXXXXXXX .

- Source code

```
package main

import (
    "fmt"
)

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

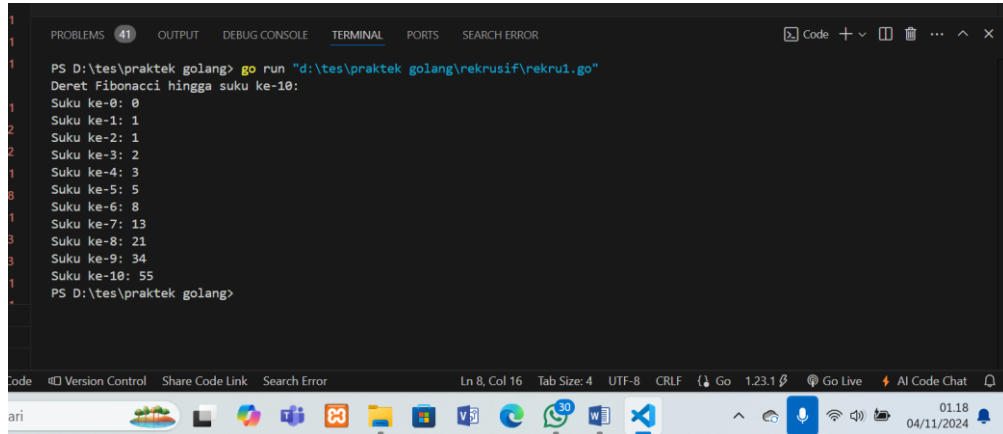
func main() {
    n := 10
```

```

    fmt.Printf("Deret Fibonacci hingga suku ke-%d:\n", n)
    for i := 0; i <= n; i++ {
        fmt.Printf("Suku ke-%d: %d\n", i, fibonacci(i))
    }
}

```

- Screenshot hasil



```

PS D:\tes\praktek golang> go run "d:\tes\praktek golang\rekursif\rekru1.go"
Deret Fibonacci hingga suku ke-10:
Suku ke-0: 0
Suku ke-1: 1
Suku ke-2: 1
Suku ke-3: 2
Suku ke-4: 3
Suku ke-5: 5
Suku ke-6: 8
Suku ke-7: 13
Suku ke-8: 21
Suku ke-9: 34
Suku ke-10: 55
PS D:\tes\praktek golang>

```

- Deskripsi program

Fungsi fibonacci menggunakan metode rekursif untuk menghitung nilai Fibonacci pada indeks tertentu, dengan mendefinisikan bahwa jika n lebih kecil atau sama dengan 1, maka nilainya adalah n itu sendiri. Dalam fungsi main, program mengatur variabel n menjadi 10 dan mencetak deret Fibonacci dari suku ke-0 hingga ke-10. Setiap hasil dari fungsi fibonacci dicetak dengan format yang menunjukkan nomor suku dan nilai Fibonacci pada suku tersebut.

2. Unguided 2

Studi case

XXXXXXXXXXXXXXXXXX

- Source code

```

package main

import "fmt"

func cetakbintang(win int) {
    if win <= 0 {
        return
    }
    cetakbintang(win - 1)
    for i := 0; i < win; i++ {
        fmt.Print("* ")
    }
    fmt.Println()
}

func main() {

```

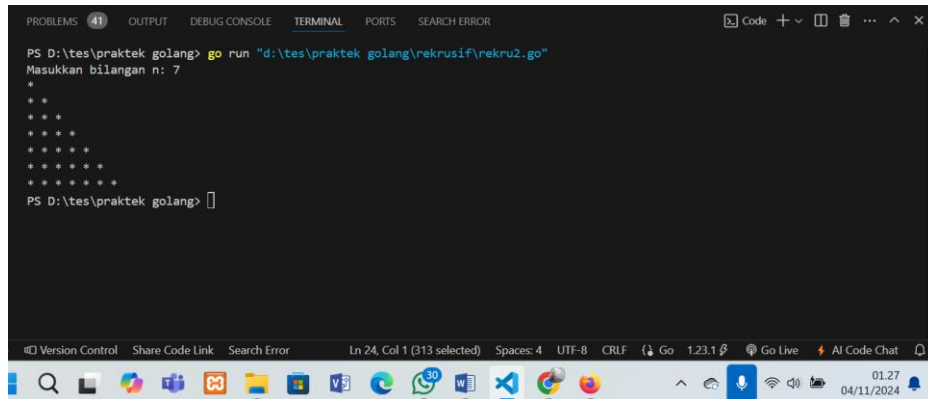
```

var win int
fmt.Print("Masukkan bilangan n: ")
fmt.Scan(&win)

fmt.Print("")
cetakbintang(win)
}

```

- Screenshot hasil



- Deskripsi program

Fungsi cetakbintang menggunakan rekursi untuk mencetak bintang secara bertahap dari 1 hingga win baris. Jika win lebih kecil atau sama dengan 0, fungsi akan berhenti. Dalam setiap pemanggilan fungsi, cetakbintang akan mencetak sejumlah bintang sesuai dengan nilai win saat itu, dengan tiap bintang dipisahkan spasi. Di fungsi main, pengguna diminta memasukkan nilai win, yang kemudian digunakan sebagai jumlah baris untuk pola bintang yang akan dicetak.

3. Unguided 3

Studi case

XXXXXXX

- Source code

```

package main

import (
    "fmt"
)

func findFactors(N, divisor int) {

    if divisor > N {
        return
    }

    if N%divisor == 0 {
        fmt.Printf("%d ", divisor)
    }
}

```



```

        findFactors(N, divisor+1)
    }

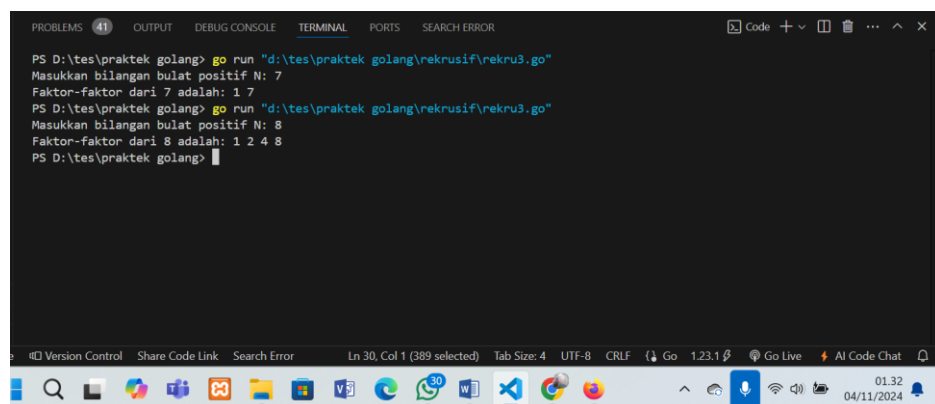
    func main() {
        var N int

        fmt.Print("Masukkan bilangan bulat positif N: ")
        fmt.Scan(&N)

        fmt.Printf("Faktor-faktor dari %d adalah: ", N)
        findFactors(N, 1)
        fmt.Println()
    }
}

```

- Screenshoot hasil



```

PS D:\tes\praktek golang> go run "d:\tes\praktek golang\rekursif\rekrus3.go"
Masukkan bilangan bulat positif N: 7
Faktor-faktor dari 7 adalah: 1 7
PS D:\tes\praktek golang> go run "d:\tes\praktek golang\rekursif\rekrus3.go"
Masukkan bilangan bulat positif N: 8
Faktor-faktor dari 8 adalah: 1 2 4 8
PS D:\tes\praktek golang>

```

- Deskripsi program

Fungsi findFactors bekerja secara rekursif dengan dua parameter, yaitu N (bilangan yang faktor-faktornya dicari) dan divisor (pembagi saat ini). Jika divisor lebih besar dari N, fungsi berhenti. Dalam setiap pemanggilan, program memeriksa apakah N habis dibagi oleh divisor. Jika ya, divisor dicetak sebagai faktor dari N. Fungsi findFactors kemudian dipanggil lagi dengan nilai divisor yang dinaikkan satu. Di main, pengguna diminta memasukkan nilai N, yang kemudian diproses oleh fungsi findFactors untuk menemukan semua faktor dari N.

4. Unguided 4

Studi case

XXXXXXXXXXXX

- Source code

```

package main

import (
    "fmt"
)

```

```

func printSeries(N, current int) {
    if current == 1 {
        fmt.Printf("%d ", current)
        return
    }

    fmt.Printf("%d ", current)
    printSeries(N, current-1)
    fmt.Printf("%d ", current)
}

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

    fmt.Printf("Barisan bilangan dari %d hingga 1 dan kembali ke
%d:\n", N, N)
    printSeries(N, N)
    fmt.Println()
}

```

- Screenshot hasil

```

PS D:\tes\praktek golang> go run "d:\tes\praktek golang\rekursif\rekr4.go"
Masukkan bilangan bulat positif N: 5
Barisan bilangan dari 5 hingga 1 dan kembali ke 5:
5 4 3 2 1 2 3 4 5
PS D:\tes\praktek golang> go run "d:\tes\praktek golang\rekursif\rekr4.go"
Masukkan bilangan bulat positif N: 9
Barisan bilangan dari 9 hingga 1 dan kembali ke 9:
9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9
PS D:\tes\praktek golang>

```

- Deskripsi Program

Program akan mencetak pola bilangan menurun dari N hingga 1 dan kemudian kembali naik ke N. Fungsi `printSeries` adalah fungsi rekursif yang menerima dua parameter: `N` (batas bilangan awal) dan `current` (bilangan yang sedang dicetak). Jika `current` sama dengan 1, program mencetak angka 1 dan menghentikan proses rekursi untuk bagian menurun. Jika `current` lebih dari 1, program mencetak `current`, lalu memanggil dirinya sendiri dengan `current` dikurangi satu untuk mencetak pola menurun. Setelah mencapai 1, fungsi melanjutkan untuk mencetak kembali `current` hingga kembali ke nilai awal `N`, membentuk pola simetris. Di fungsi `main`, pengguna diminta memasukkan nilai `N`, dan `printSeries` kemudian mencetak pola sesuai input tersebut.

5. Unguided 5

Studi case

XXXXXXXXXX

- Source code

```
package main

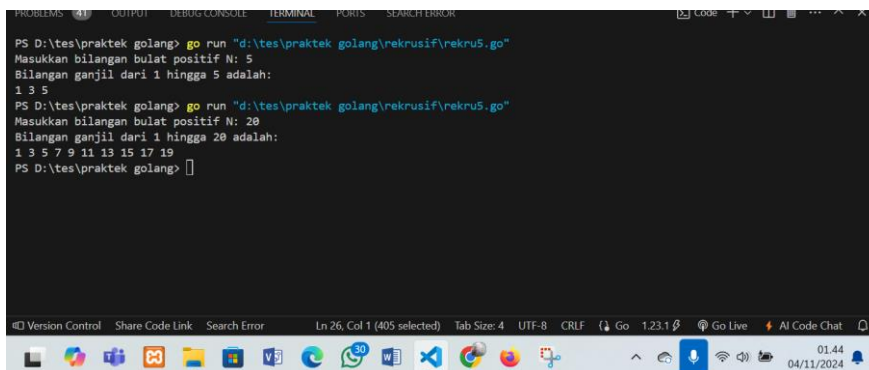
import (
    "fmt"
)

func printOddNumbers(N, current int) {
    if current > N {
        return
    }
    if current%2 != 0 {
        fmt.Printf("%d ", current)
    }
    printOddNumbers(N, current+1)
}

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

    fmt.Printf("Bilangan ganjil dari 1 hingga %d adalah:\n", N)
    printOddNumbers(N, 1)
    fmt.Println()
}
```

- Screenshot hasil



```
PS D:\tes\praktek golang> go run "d:\tes\praktek golang\rekursif\rekru5.go"
Masukkan bilangan bulat positif N: 5
Bilangan ganjil dari 1 hingga 5 adalah:
1 3 5
PS D:\tes\praktek golang> go run "d:\tes\praktek golang\rekursif\rekru5.go"
Masukkan bilangan bulat positif N: 20
Bilangan ganjil dari 1 hingga 20 adalah:
1 3 5 7 9 11 13 15 17 19
PS D:\tes\praktek golang>
```

- Deskripsi Program

Program akan mencetak semua bilangan ganjil dari 1 hingga bilangan N yang dimasukkan oleh pengguna. Fungsi printOddNumbers menggunakan rekursi dengan dua parameter: N (batas bilangan akhir) dan current (bilangan saat ini yang sedang diperiksa). Jika current lebih besar dari N, fungsi akan berhenti. Jika

current merupakan bilangan ganjil (diperiksa dengan `current%2 != 0`), maka current dicetak. Fungsi kemudian memanggil dirinya sendiri dengan current ditambah satu untuk melanjutkan ke bilangan berikutnya. Di fungsi main, pengguna diminta memasukkan nilai N, lalu `printOddNumbers` mencetak semua bilangan ganjil dari 1 hingga N.

6. Unguided 6

Studi case

XXXXXXXXXXXX

- Source case

```
package main

import "fmt"

func pangkat(x, y int) int {
    if y == 0 {
        return 1
    }

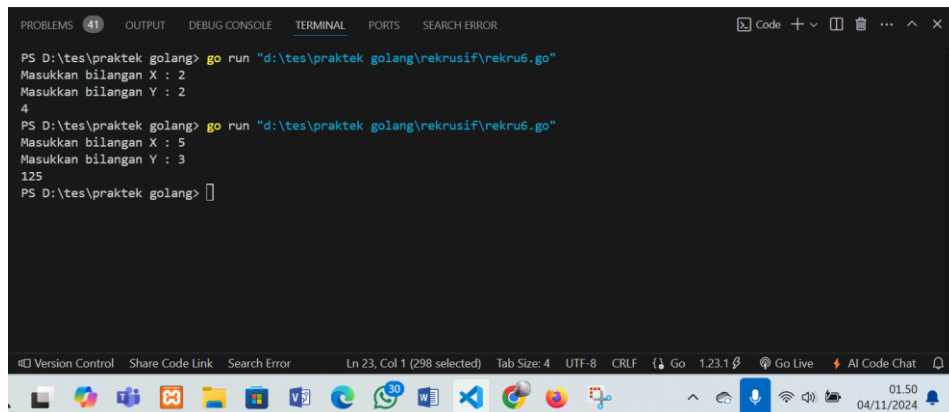
    return x * pangkat(x, y-1)
}

func main() {
    var x, y int

    fmt.Print("Masukkan bilangan X : ")
    fmt.Scan(&x)
    fmt.Print("Masukkan bilangan Y : ")
    fmt.Scan(&y)

    fmt.Print(pangkat(x, y))
}
```

- Screenshoot hasil



```
PS D:\tes\praktek_golang> go run "d:\tes\praktek_golang\rekursif\rekr6.go"
Masukkan bilangan X : 2
Masukkan bilangan Y : 2
4
PS D:\tes\praktek_golang> go run "d:\tes\praktek_golang\rekursif\rekr6.go"
Masukkan bilangan X : 5
Masukkan bilangan Y : 3
125
PS D:\tes\praktek_golang>
```

- Deskripsi Program

Program memiliki fungsi pangkat yang menggunakan rekursi untuk menghitung x pangkat y , di mana x adalah bilangan yang akan dipangkatkan dan y adalah eksponen/pangkatnya. Di dalam fungsi main, program meminta pengguna untuk memasukkan dua bilangan - X sebagai bilangan yang akan dipangkatkan dan Y sebagai eksponen, kemudian menampilkan hasil perpangkatan menggunakan fungsi pangkat.