LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK

MODUL 5

REKURSIF

Oleh:

DAMARA GALUH PEMBAYUN

2311102110

IF-11-02

S1 TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

I. DASAR TEORI

Metode pemecahan masalah regresif melibatkan fungsi yang memanggil dirinya sendiri secara langsung atau tidak langsung. Konsep ini sangat kuat dalam pemrograman karena memungkinkan kita untuk memecahkan masalah kompleks dengan cara yang lebih halus dan seringkali lebih mudah dipahami.

Rekursif adalah alat yang kuat dalam kotak alat seorang programmer. Dengan memahami konsep dasar dan kapan harus menggunakannya, Anda dapat menulis kode yang lebih elegan dan efisien. Namun, perlu diingat bahwa rekursi tidak selalu merupakan solusi terbaik untuk setiap masalah. Pertimbangkan trade-off antara kemudahan pemahaman, performansi, dan penggunaan memori saat memilih antara rekursi dan iterasi.

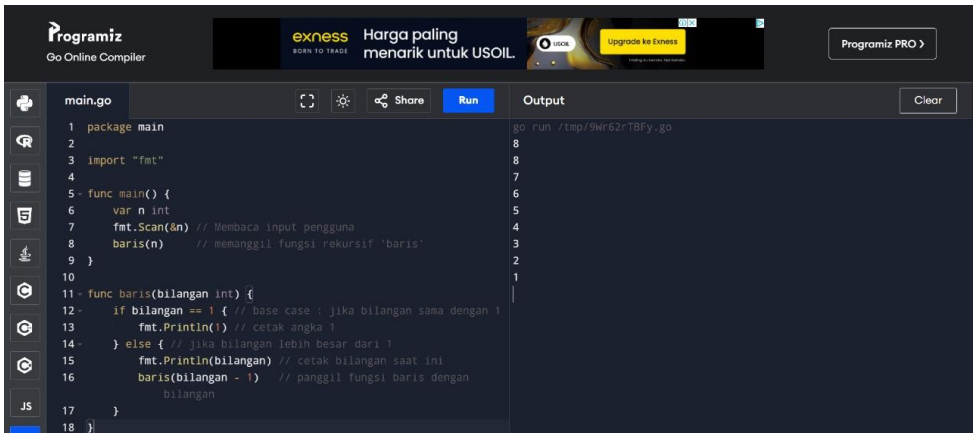
II. GUIDED

Guided 1

SourchCode

```
package main

import "fmt"

func main (){
    var int
    fmt.Scan (&n) //membaca input pengguna
    baris (n) //memanggil fungsi reskrusif 'baris'
}
func baris (bilangan int){
    if bilangan == 1 { //basecase apabila bilangan sama
dengan 1
        fmt.Println (1) //cetak angka 1
    } else { //jika bilangan lebih besar dari 1
        fm.Prinln (bilangan) //cetak bilangan saat ini
        baris (nilangan-1) //panggil fungsi baris dengan
bilangan 1
    }
}
```

Hasil ScreenShoot

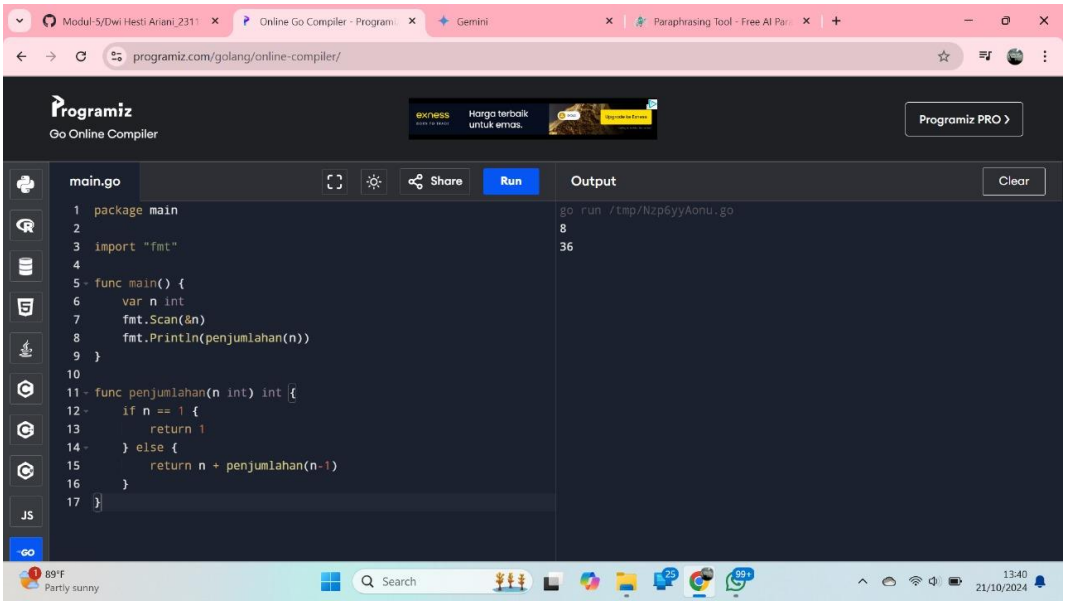


Guided 2

SourchCode

```
import "fmt"
func main (){
    var int
    fmt.Scan (&n)
    fmt.Println (penjumlahan(n))
}
      func penjumlahan (n int)int {
        if n == 1 {
            return 1
        }else{
            return n + penjumlahan (n-1)
        }
      }
```

Hasil ScreenShoot



III. UNGUIDED

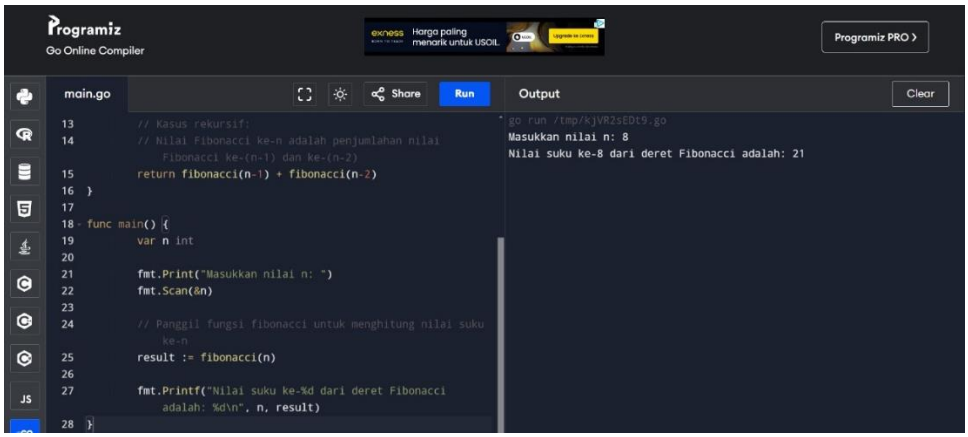
Unguided 1

SourchCode

```
package main

import "fmt"

// fibonacci adalah fungsi rekursif untuk menghitung nilai
suku ke-n dari deret Fibonacci
func fibonacci(n int) int {
        // Kasus dasar:
        // Jika n adalah 0 atau 1, nilai Fibonacci-nya
adalah n itu sendiri
        if n <= 1 {
                return n
        }

        // Kasus rekursif:
        // Nilai Fibonacci ke-n adalah penjumlahan nilai
Fibonacci ke-(n-1) dan ke-(n-2)
        return fibonacci(n-1) + fibonacci(n-2)
}

func main() {
        var n int

        fmt.Print("Masukkan nilai n: ")
        fmt.Scan(&n)
```

```
        // Panggil fungsi fibonacci untuk menghitung nilai
suku ke-n
        result := fibonacci(n)

        fmt.Printf("Nilai suku ke-%d dari deret Fibonacci
adalah: %d\n", n, result)
}
```

Hasil ScreenShoot



Deskripsi

Ini adalah program Go yang dimaksudkan untuk menghitung nilai suku ke-n dari deret Fibonacci dengan menggunakan teknik rekursif. Salah satu contoh deret Fibonacci adalah urutan angka di mana setiap angka adalah jumlah dari dua angka sebelumnya. Contohnya adalah 0, 1, 1, 2, 3, 5, 8, 13, dan seterusnya.

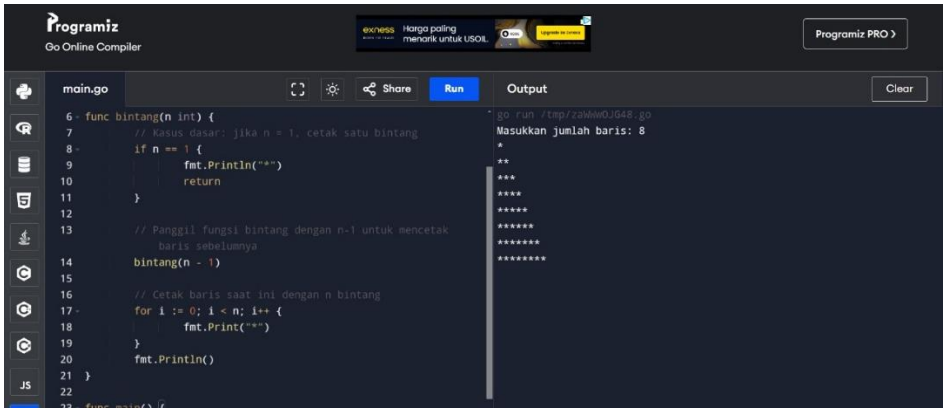
Unguided 2

SourchCode

```
package main

import "fmt"

// bintang adalah fungsi rekursif untuk mencetak pola
bintang
func bintang(n int) {
        // Kasus dasar: jika n = 1, cetak satu bintang
        if n == 1 {
                fmt.Println("*")
                return
        }

        // Panggil fungsi bintang dengan n-1 untuk
mencetak baris sebelumnya
        bintang(n - 1)
```

```
        // Cetak baris saat ini dengan n bintang
        for i := 0; i < n; i++ {
                fmt.Print("*")
        }
        fmt.Println()
}

func main() {
        var n int

        fmt.Print("Masukkan jumlah baris: ")
        fmt.Scan(&n)

        // Panggil fungsi bintang untuk mencetak pola
        bintang(n)
}
```

Hasil ScreenShoot



Deskripsi

Dengan menggunakan pendekatan rekursif, program Go ini dimaksudkan untuk mencetak pola segitiga siku-siku yang terdiri dari tanda bintang (*). Setiap baris pola memiliki jumlah bintang yang sama dengan nomor barisnya.

Masalah dibagi menjadi submasalah yang lebih kecil dengan regresi pada fungsi bintang. Setiap kali fungsi dipanggil, masalahnya menjadi mencetak satu baris bintang yang lebih pendek.

Misalnya, jika kita ingin mencetak pola yang terdiri dari empat baris, maka:

1.bintang(4) akan memanggil bintang(3) untuk mencetak 3 baris pertama.

2.bintang(3) akan memanggil bintang(2) untuk mencetak 2 baris pertama.

3.bintang(2) akan memanggil bintang(1) untuk mencetak 1 baris pertama.

4.bintang(1) adalah kasus dasar, sehingga hanya mencetak satu bintang.

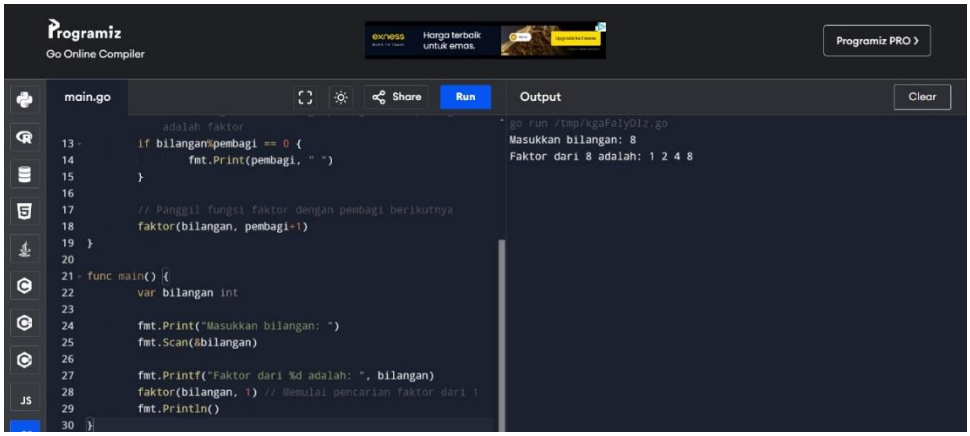
5.Setelah bintang(1) selesai, program kembali ke bintang(2) dan mencetak baris kedua dengan 2 bintang.
6.Proses berlanjut hingga semua baris selesai dicetak.

Unguided 3

SourchCode

```
package main

import "fmt"

// faktor adalah fungsi rekursif untuk mencari faktor dari
suatu bilangan
func faktor(bilangan, pembagi int) {
        // Kasus dasar: jika pembagi lebih besar dari
bilangan, tidak ada faktor lagi
        if pembagi > bilangan {
                return
        }

        // Jika bilangan habis dibagi pembagi, maka
pembagi adalah faktor
        if bilangan%pembagi == 0 {
                fmt.Print(pembagi, " ")
        }

        // Panggil fungsi faktor dengan pembagi berikutnya
        faktor(bilangan, pembagi+1)
}

func main() {
        var bilangan int

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

        fmt.Printf("Faktor dari %d adalah: ", bilangan)
        faktor(bilangan, 1) // Memulai pencarian faktor
dari 1
        fmt.Println()
}
```

Hasil ScreenShoot



Deskripsi

Ini adalah aplikasi yang menunjukkan cara menggunakan rekursi dalam bahasa Go untuk mencari faktor dari suatu bilangan. Meskipun regresi adalah alat pemrograman yang kuat, itu harus digunakan dengan hati-hati agar tidak menyebabkan masalah kinerja. Dalam hal ini, rekursi adalah pendekatan yang ideal karena struktur masalahnya sesuai dengan konsep rekursi.

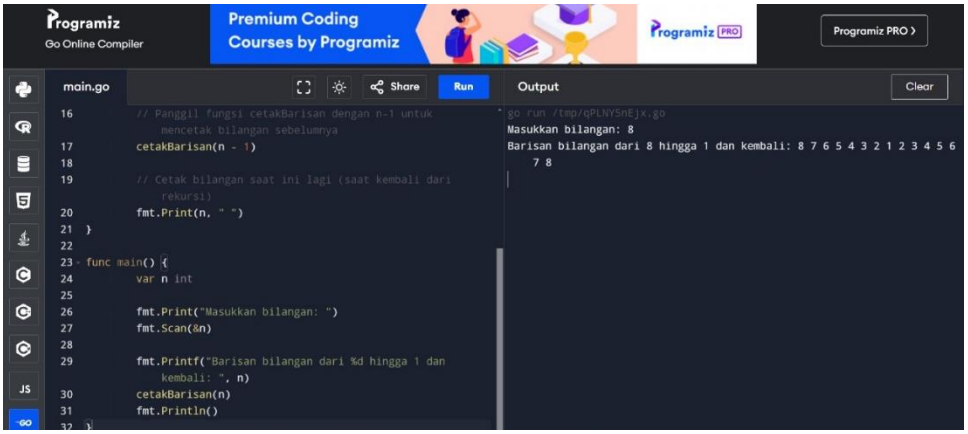
Unguided 4

SourchCode

```
package main

import "fmt"

// cetakBarisan adalah fungsi rekursif untuk mencetak
barisan bilangan
func cetakBarisan(n int) {
        // Kasus dasar: jika n = 1, cetak 1
        if n == 1 {
                fmt.Print(n, " ")
                return
        }

        // Cetak bilangan saat ini
        fmt.Print(n, " ")

        // Panggil fungsi cetakBarisan dengan n-1 untuk
mencetak bilangan sebelumnya
        cetakBarisan(n - 1)

        // Cetak bilangan saat ini lagi (saat kembali dari
rekursi)
        fmt.Print(n, " ")
}
```

```
func main() {
        var n int

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

        fmt.Printf("Barisan bilangan dari %d hingga 1 dan
kembali: ", n)
        cetakBarisan(n)
        fmt.Println()
}
```

Hasil ScreenShoot



Deskripsi

Contoh sederhana penggunaan rekursi dalam bahasa Go untuk mencetak barisan bilangan dengan pola naik-turun tersedia dalam program ini. Meskipun regresi adalah alat pemrograman yang kuat, itu harus digunakan dengan hati-hati agar tidak menyebabkan masalah kinerja. Dalam hal ini, rekursi adalah pendekatan yang ideal karena struktur masalahnya sesuai dengan konsep rekursi.

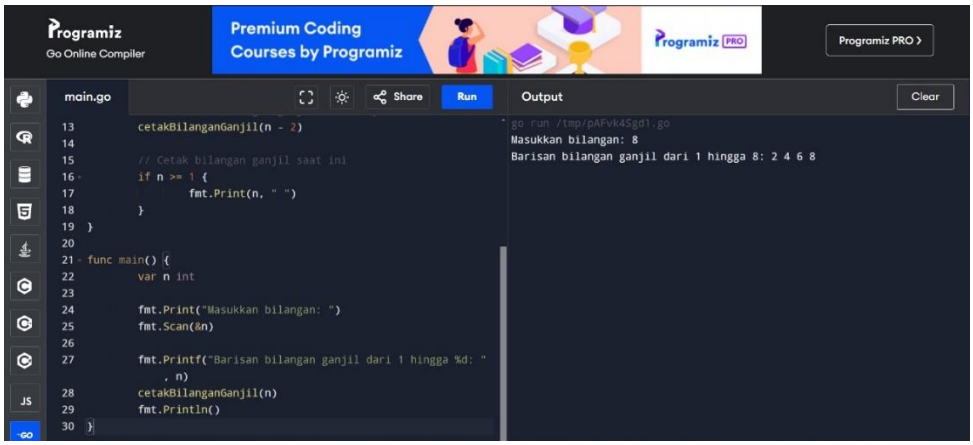
Unguided 5

SourchCode

```
package main

import "fmt"

// cetakBilanganGanjil adalah fungsi rekursif untuk
mencetak bilangan ganjil
func cetakBilanganGanjil(n int) {
```

```
        // Kasus dasar: jika n kurang dari 1, tidak ada
bilangan ganjil yang perlu dicetak
        if n < 1 {
                return
        }

        // Panggil fungsi cetakBilanganGanjil dengan n-2
untuk mencetak bilangan ganjil sebelumnya
        cetakBilanganGanjil(n - 2)

        // Cetak bilangan ganjil saat ini
        if n >= 1 {
                fmt.Print(n, " ")
        }
}

func main() {
        var n int

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

        fmt.Printf("Barisan bilangan ganjil dari 1 hingga
%d: ", n)
        cetakBilanganGanjil(n)
        fmt.Println()
}
```

Hasil ScreenShoot



Deskripsi

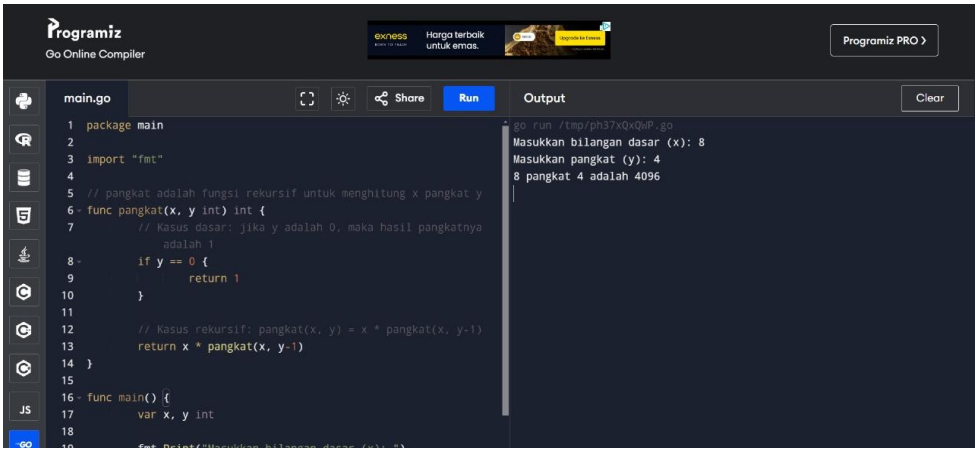
Program ini memberikan contoh sederhana tentang penggunaan rekursi dalam bahasa Go untuk mencetak barisan bilangan ganjil. Rekursi adalah alat yang kuat dalam pemrograman, tetapi perlu digunakan dengan bijak agar tidak menyebabkan masalah kinerja. Dalam kasus ini, rekursi adalah

pilihan yang baik karena struktur masalahnya sangat cocok dengan konsep rekursif.

Unguided 6
SourchCode

```
package main

import "fmt"

// pangkat adalah fungsi rekursif untuk menghitung x
pangkat y
func pangkat(x, y int) int {
        // Kasus dasar: jika y adalah 0, maka hasil
pangkatnya adalah 1
        if y == 0 {
                return 1
        }

        // Kasus rekursif: pangkat(x, y) = x * pangkat(x,
y-1)
        return x * pangkat(x, y-1)
}

func main() {
        var x, y int

        fmt.Print("Masukkan bilangan dasar (x): ")
        fmt.Scan(&x)
        fmt.Print("Masukkan pangkat (y): ")
        fmt.Scan(&y)

        hasil := pangkat(x, y)
        fmt.Printf("%d pangkat %d adalah %d\n", x, y,
hasil)
}
```

Hasil ScreenShoot



Deskripsi

Program Go ini dirancang untuk menghitung nilai pangkat dari suatu bilangan secara rekursif. Rekursi adalah teknik pemrograman di mana sebuah fungsi memanggil dirinya sendiri untuk memecahkan masalah yang lebih kecil. Dalam konteks perhitungan pangkat, rekursi digunakan untuk memecah masalah menjadi perkalian berulang.