LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 5

REKURSIF

Oleh:

MUHAMMAD DAFFA AL FAIZ

2311102237

S1IF-11-02

S1 TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

I. DASAR TEORI

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() diatas, terlihat pada bari 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 subprogam cetak() nilai x akan selalu bertambah 1 (increment by one) secara terus menerus tanpa henti

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

```
D:\DEV\DEMO>go build contoh.go
D:\DEV\DEMO>contoh.exe
5
6
7
8
9
10
11
12
13
...
```

Oleh karena itu bisanya ditambahkan struktur kontrol 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

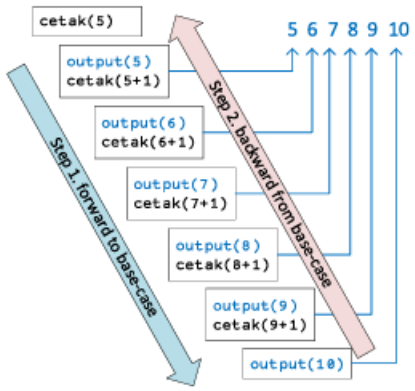
```
procedure cetak(in x:integer)
algoritma
    if x == 10 then
        output(x)
    else
        output(x)
        cetak(x+1)
    endif
endprocedure
```

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 tejadi. Kondisi dari **recursive-case** ini adalah negasi dari kondisi **base-case** atau ketika nilai x != 10.

```go
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)
    }
}
```

```
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 berhenti pada saat kondisi **base-case** terpenuhi atau **true.** Setelah itu akan terjadi proses **backward** atau kembali ke subprogram yang sebelumnya. Artinya setelah semua instruRsi **cetak{lO)** selesai dieksekusi, maka program akan kembali ke **cetak{9)** yang memanggil cetak(lO) tersebut. Begitu seterusnya hingga kembali ke cetak(S).

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

Telkom Universitu

```
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:

- TelmiR reRursif ini merupaRan salah satu alternatif untuR mengganti struRtur Rontrol perulangan dengan memanfaatRan subprogram (bisa fungsi ataupun prosedure).
- UntuR menghentiRan proses reRursif digunaRan percabangan (if-then).
- **Base-case** adalah Rondisi proses reRursif berhenti. **Base-case** merupaRan hal terpenting dan pertama yang harus diRetahui RetiRa aRan membuat program reRursif. **Mustahil** membuat program reRursif tanpa mengetahui **base-case** terlebih dahulu.
- **Recursive-case** adalah Rondisi dimana proses pemanggilan dirinya sendiri dilaRuRan. Kondisi **recursive-case** adalah Romplemen atau negasi dari **base-case.**
- Setiap algoritma reRursif selalu memiliRi padanan dalam bentuR algoritma interatif.

Komponen rekursif

Algoritma rekursif terdiri dari dua komponen utama:

- Base-case(basis), yaitu bagian untuk menghentikan proses rekursif dan menjadi komponen terpenting di dalam sebuah rekursif
- Rekursif-case, yaitu bagian pemanggilan subprogramnya

Contoh program dengan menggunakan rekursif

a. Membuat baris bilangan dari n hingga 1
   Base-case: bilangan == 1

```
1  package main
2  import "fmt"
3  func main(){
4      var n int
5      fmt.Scan(&n)
6      baris(n)
7  }
8
9  func baris(bilangan int){
10     if bilangan == 1 {
11         fmt.Println(1)
12     }else{
13         fmt.Println(bilangan)
14         baris(bilangan - 1)
15     }
16 }
```

b. Menghitung hasil penjumlahan 1 hingga n
   Base-case: n == 1

```
1  package main
2  import "fmt"
3  func main(){
4      var n int
5      fmt.Scan(&n)
6      fmt.Println(penjumlahan(n))
7  }
8
9  func penjumlahan(n int) int {
10     if n == 1 {
11         return 1
12     }else{
13         return n + penjumlahan(n-1)
14     }
15 }
```

c. Mencari dua pangkat n atau 2n
   Base-case: n == 0

```
package main
import "fmt"
func main(){
    var n int
    fmt.Scan(&n)
    fmt.Println(pangkat(n))
}

func pangkat(n int) int {
    if n == 0 {
        return 1
    }else{
        return 2 * pangkat(n-1)
    }
}
```

d. Mencari nilai faktorial atau n!
   Base-case: n == 0 atau n == 1

```
package main
import "fmt"
func main(){
    var n int
    fmt.Scan(&n)
    fmt.Println(faktorial(n))
}

func faktorial(n int) int {
    if n == 0 || n == 1 {
        return 1
    }else{
        return n * faktorial(n-1)
    }
}
```

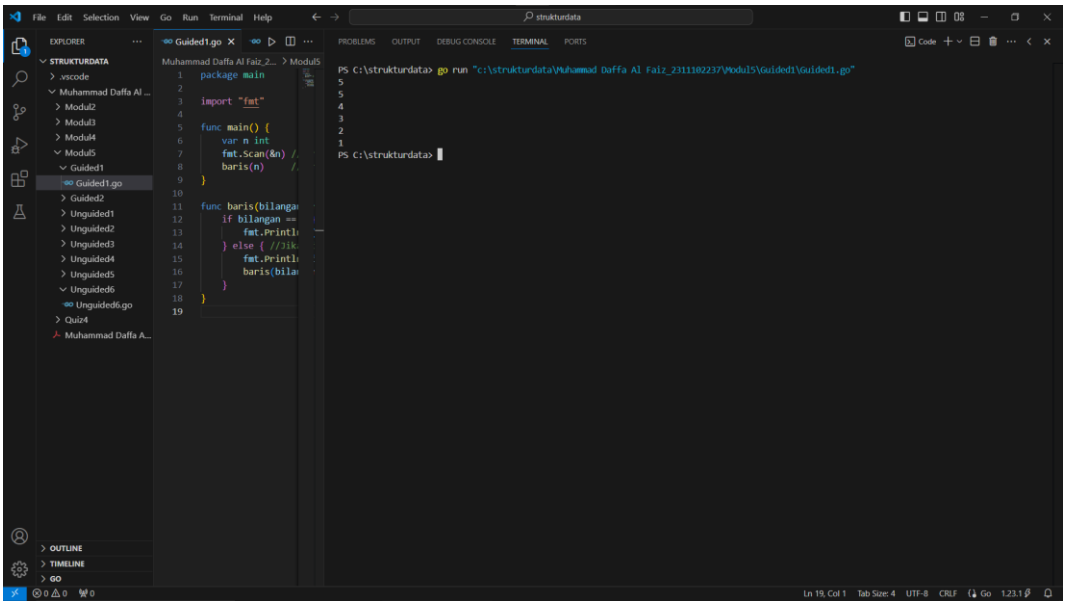
II. GUIDED

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 { //Base case: jika bilangan sama dengan
        fmt.Println(1) //Cetak angka
    } else { //Jika bilangan lebih besar dari
        fmt.Println(bilangan) //Cetak bilanga saat ini
        baris(bilangan - 1)   //Panggil fungsi "baris" dengan
    }
}
```

Screenshot



Deskripsi

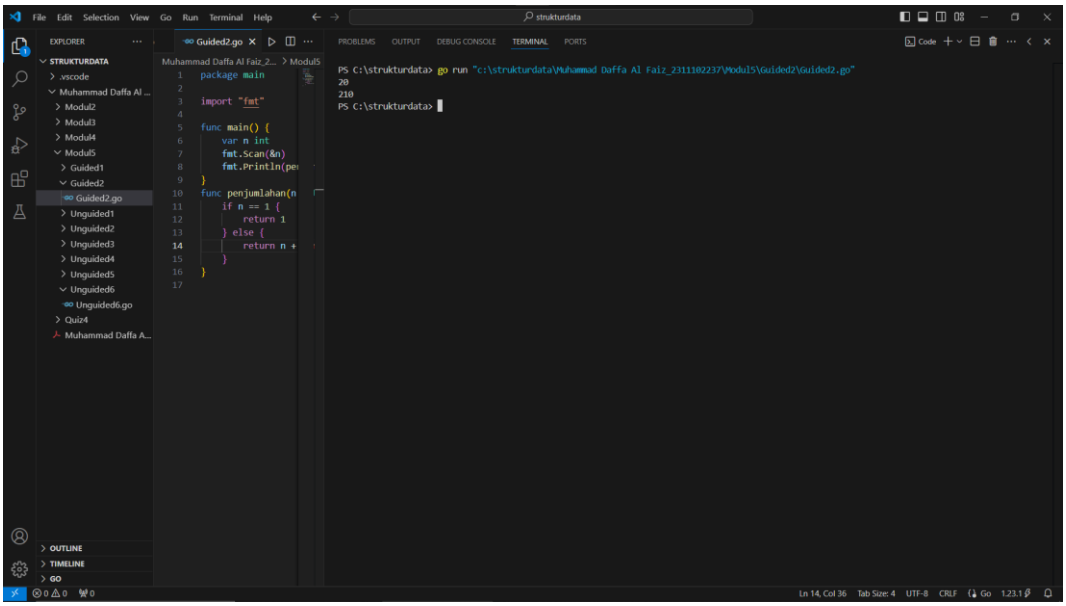
Program ini adalah untuk mencetak bilangan dari input pengguna secara menurun hingga 1

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



Deskripsi

Program ini adalah untuk menghitung jumlah dari semua bilangan bulat dari 1 hingga n, Dimana n adalah angka yang dimasukan oleh pengguna.

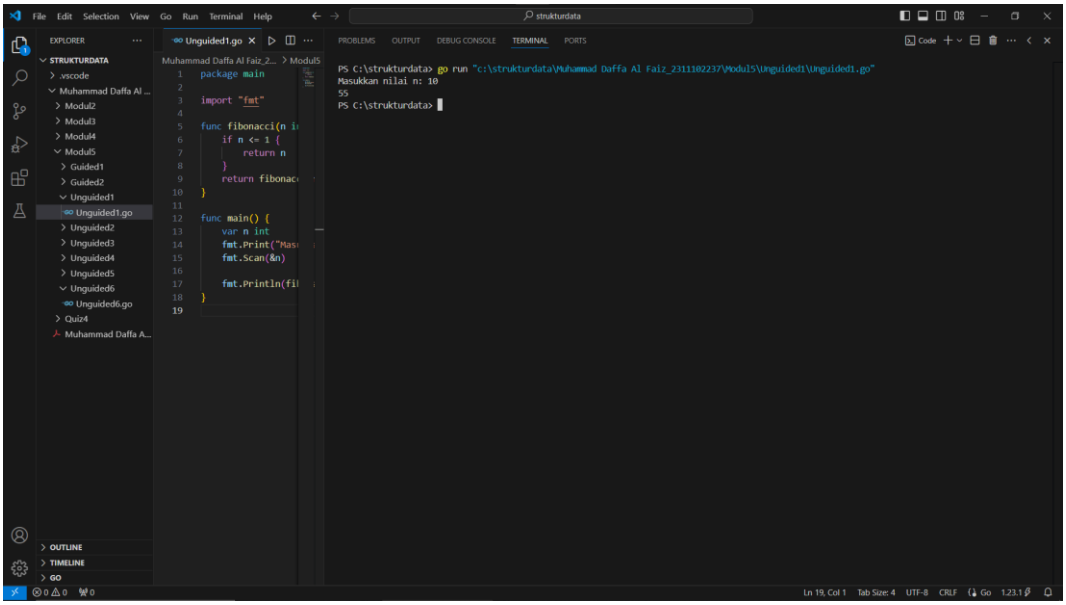
III. UNGUIDED

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() {
    var n int
    fmt.Print("Masukkan nilai n: ")
    fmt.Scan(&n)

    fmt.Println(fibonacci(n))
}
```

Screenshot



Deskripsi

Program ini adalah untuk menghitung dan mencetak sukuk e-n dari deret Fibonacci. Fungsi Fibonacci menggunakan rekursi untuk menghitung suku suku sebelumnya secara berulang.

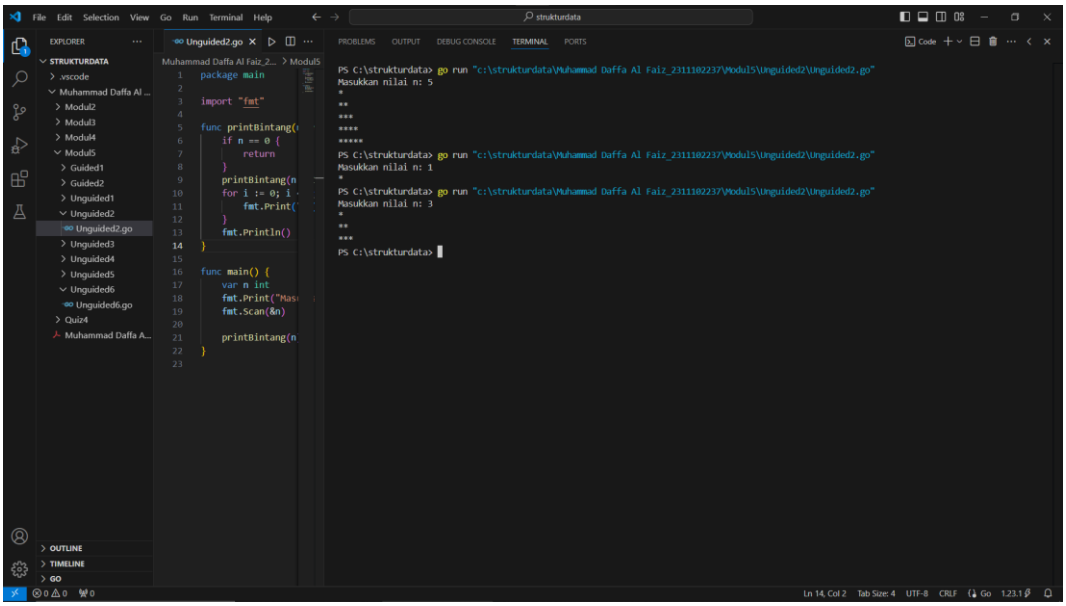
Unguided 2

Source Code

```
package main

import "fmt"

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

func main() {
    var n int
```

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

    printBintang(n)
}
```

Screenshot



Deskripsi

Program ini adalah untuk mencetak pola piramida Bintang dengan ketinggian n, Dimana n adalah bilangan bulat yang dimasukan oleh pengguna. Fungsi printBintang menggunakan rekursi untuk mencetak baris baris piramida secara berulang, dengan setiap baris berisi sejumlah Bintang sesuai dengan nomor barisannya.

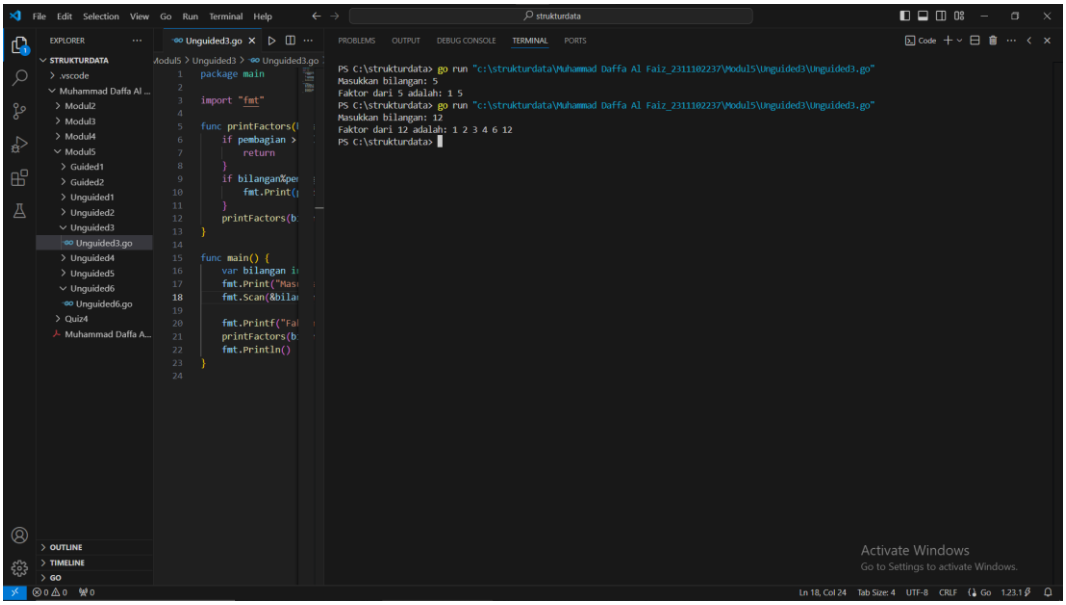
Unguided 3

Source Code

```
package main

import "fmt"

func printFactors(bilangan, pembagian int) {
    if pembagian > bilangan {
```

```
        return
    }
    if bilangan%pembagian == 0 {
        fmt.Print(pembagian, " ")
    }
    printFactors(bilangan, pembagian+1)
}

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

    fmt.Printf("Faktor dari %d adalah: ", bilangan)
    printFactors(bilangan, 1)
    fmt.Println()
}
```

Screenshot



Deskripsi

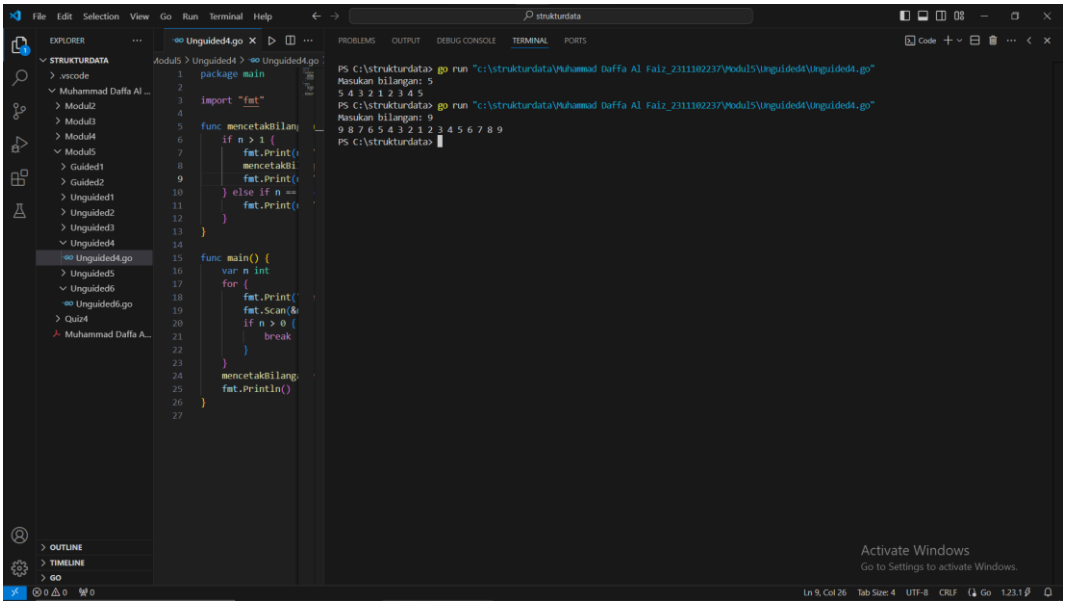
Program ini adalah untuk mencetak semua factor dari sebuah bilangan bulat yang dimasukan oleh pengguna. Fungsi printFactors menggunakan rekursi untuk memeriksa setaip bilangan mulai dari 1 hingga bilangan itu sendiri.

Unguided 4

Source Code

```
package main

import "fmt"

func mencetakBilangan(n int) {
    if n > 1 {
        fmt.Print(n, " ")
        mencetakBilangan(n - 1)
        fmt.Print(n, " ")
    } else if n == 1 {
        fmt.Print(n, " ")
    }
}

func main() {
    var n int
    for {
        fmt.Print("Masukan bilangan: ")
        fmt.Scan(&n)
        if n > 0 {
            break
        }
    }
    mencetakBilangan(n)
    fmt.Println()
}
```

Sreenshot



Deskripsi

Program ini adalah untuk mencetak bilangan dari 1 hingga n dan kemudian Kembali mencetaknya dalam urutan terbalik. Fungsi mencetakBilangan Menggunakan rekursi untuk mencetak bilangan secara naik dan kemudian turun.

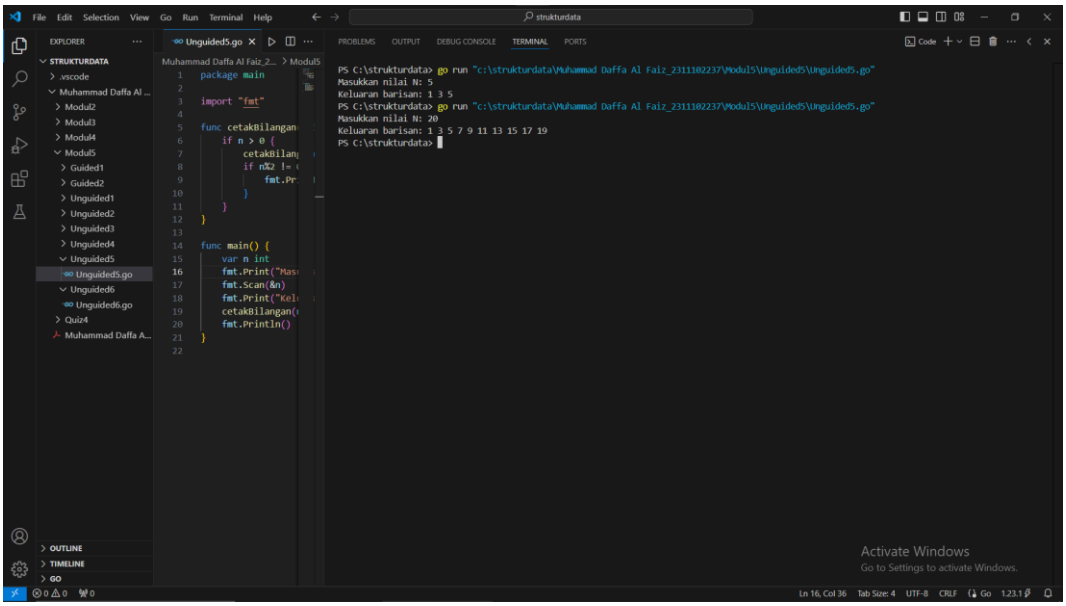
Unguided 5

Source Code

```
package main

import "fmt"

func cetakBilangan(n int) {
    if n > 0 {
        cetakBilangan(n - 1)
        if n%2 != 0 {
            fmt.Printf("%d ", n)
        }
    }
}

func main() {
    var n int
    fmt.Print("Masukkan nilai N: ")
```

```
    fmt.Scan(&n)
    fmt.Print("Keluaran barisan: ")
    cetakBilangan(n)
    fmt.Println()
}
```

Sreenshot



Deskripsi

Program ini adalah untuk mencetak bilangan ganjil dari 1 hingga n secara rekursif. Fungsi cetakBilangan akan memanggil dirinya sendiri dengan nilai n yang terus dikurangi 1 hingga mencapai 0.

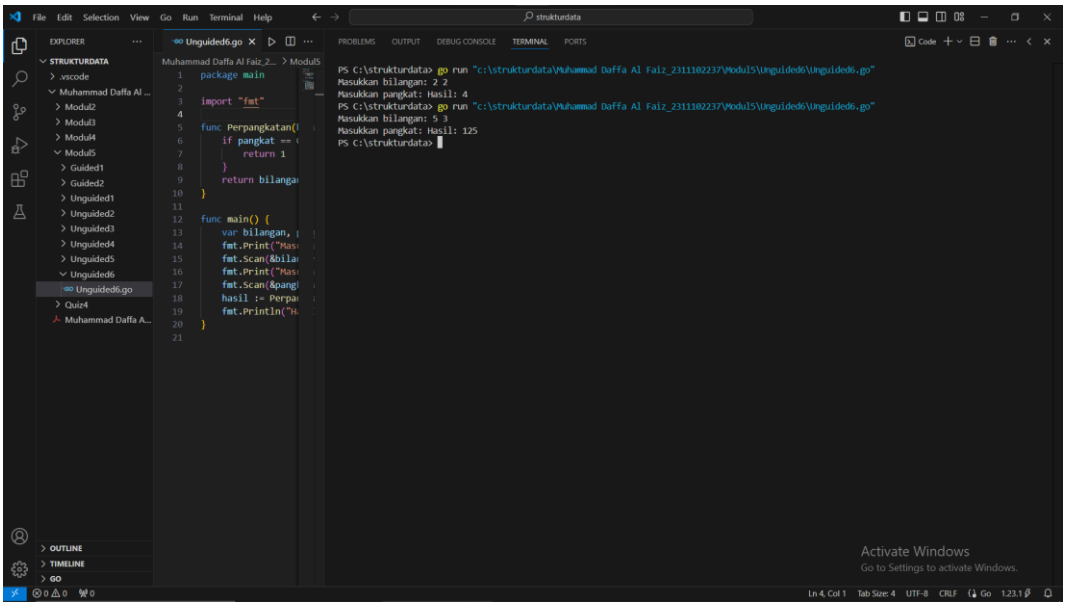
Unguided 6

Source Code

```
package main

import "fmt"

func Perpangkatan(bilangan, pangkat int) int {
    if pangkat == 0 {
        return 1
    }
```

```
    return bilangan * Perpangkatan(bilangan, pangkat-1)
}

func main() {
    var bilangan, pangkatan int
    fmt.Print("Masukkan bilangan: ")
    fmt.Scan(&bilangan)
    fmt.Print("Masukkan pangkat: ")
    fmt.Scan(&pangkatan)
    hasil := Perpangkatan(bilangan, pangkatan)
    fmt.Println("Hasil:", hasil)
}
```

Sreenshot



Deskripsi

Program ini untuk menghitung hasil perpangkatan dari suatu bilangan secara rekursif. Fungsi Perpangkatan menerima dua parameter yaitu bilangan dan pangkat