

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 5  
REKURSIF**



**Disusun Oleh :**

**HANIF REYHAN ZHAFRAN ADRYTONA / 2311102266**

**11 IF 06**

**Dosen Pengampu :**

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

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

## I. DASAR TEORI

Rekursi adalah teknik pemrograman yang memungkinkan suatu fungsi untuk memanggil dirinya sendiri guna menyelesaikan masalah yang kompleks dengan memecahnya menjadi sub-masalah yang lebih kecil dan sejenis. Proses rekursi berlangsung hingga kondisi tertentu, disebut **basis kasus**, tercapai. Basis kasus ini penting untuk mengakhiri pemanggilan fungsi agar tidak terus berjalan tanpa batas. Setiap langkah rekursi akan mendekatkan fungsi pada basis kasus, menyederhanakan masalah hingga menjadi cukup kecil untuk diselesaikan langsung. Contoh penggunaan rekursi yang sederhana adalah fungsi faktorial, yang menghitung hasil perkalian bilangan secara berurutan hingga 1.

Rekursi dapat dibagi menjadi beberapa jenis, di antaranya **rekursi langsung** (fungsi memanggil dirinya sendiri) dan **rekursi tidak langsung** (fungsi memanggil fungsi lain yang kemudian memanggil fungsi asalnya). Selain itu, ada juga rekursi **tail-end** di mana pemanggilan rekursif terjadi di akhir fungsi, dan **head recursion** di mana pemanggilan rekursif terjadi sebelum langkah lain. Keunggulan rekursi adalah kesederhanaan kode dalam menyelesaikan masalah kompleks yang dapat dipecah menjadi sub-masalah, seperti pada algoritma divide and conquer, pohon, atau grafik. Namun, kekurangannya adalah penggunaan memori yang lebih besar karena setiap pemanggilan membutuhkan tempat di call stack, dan risiko infinite loop jika basis kasus tidak ditetapkan dengan tepat.

## II. GUIDED

Guided 1.0

### Sourcecode

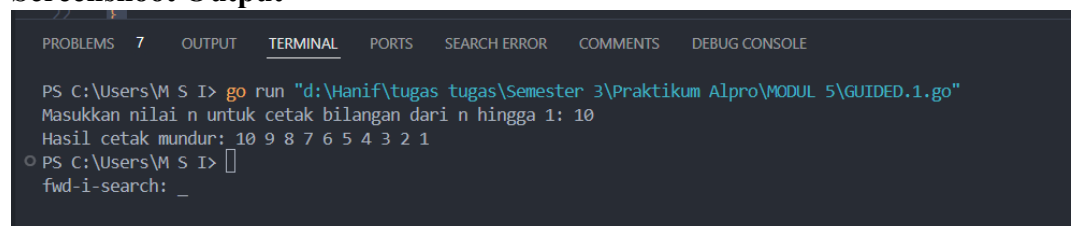
```
// 2311102266_Hanif reyhan zharan adrytona
package main

import "fmt"

// Fungsi untuk mencetak bilangan dari n hingga 1
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)
}
```

### Screenshoot Output



```
PS C:\Users\M S I> go run "d:\Hanif\tugas tugas\Semester 3\Praktikum Alpro\MODUL 5\GUIDED.1.go"
Masukkan nilai n untuk cetak bilangan dari n hingga 1: 10
Hasil cetak mundur: 10 9 8 7 6 5 4 3 2 1
PS C:\Users\M S I>
fwd-i-search: _
```

### Deskripsi Program

Program di atas merupakan contoh penerapan rekursi untuk mencetak bilangan dari nilai yang diberikan hingga angka 1 secara menurun. Fungsi utama yang menangani ini adalah cetakMundur, yang menerima parameter n sebagai batas awal bilangan yang akan dicetak mundur. Pada setiap pemanggilan fungsi, nilai n akan dicetak terlebih

dahulu dan kemudian fungsi `cetakMundur` memanggil dirinya sendiri dengan nilai  $n-1$ . Ketika  $n$  mencapai 1, fungsi mencetak angka 1 dan mengakhiri proses rekursi dengan pernyataan return, yang memastikan tidak ada lagi pemanggilan rekursif setelah kondisi dasar tersebut tercapai. Kondisi dasar (basis case) ini penting agar rekursi berhenti dan tidak memasuki loop tanpa batas.

Pada fungsi main, program meminta masukan dari pengguna untuk nilai awal  $n$ , yaitu bilangan yang menjadi awal deret cetak mundur. Program kemudian memanggil cetakMundur dengan parameter  $n$ , sehingga menghasilkan cetakan mundur dari  $n$  hingga 1. Penggunaan rekursi di sini sangat membantu dalam membuat kode lebih sederhana dan intuitif, tanpa perlu menggunakan perulangan eksplisit seperti for atau while. Hal ini menunjukkan keunggulan rekursi dalam membuat solusi yang lebih ringkas untuk masalah-masalah yang melibatkan proses pengurangan atau pengulangan hingga kondisi tertentu tercapai.

## Guided 1.2

### Sourcecode

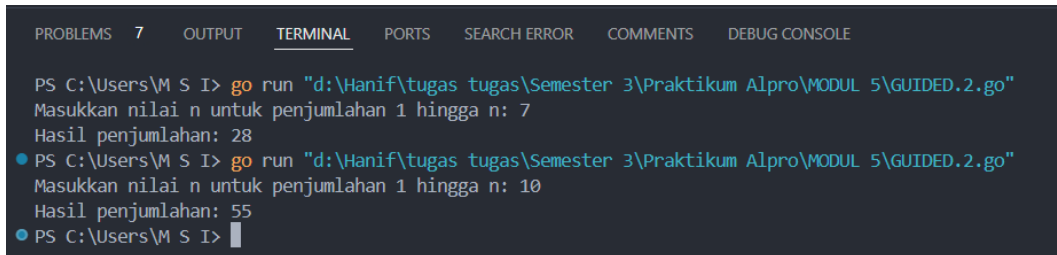
```
// 2311102266_Hanif reyhan zharan adrytona

package main
import "fmt"

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

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

## Screenshoot Output



```
PROBLEMS 7 OUTPUT TERMINAL PORTS SEARCH ERROR COMMENTS DEBUG CONSOLE

PS C:\Users\M S I> go run "d:\Hanif\tugas tugas\Semester 3\Praktikum Alpro\MODUL 5\GUIDED.2.go"
Masukkan nilai n untuk penjumlahan 1 hingga n: 7
Hasil penjumlahan: 28
● PS C:\Users\M S I> go run "d:\Hanif\tugas tugas\Semester 3\Praktikum Alpro\MODUL 5\GUIDED.2.go"
Masukkan nilai n untuk penjumlahan 1 hingga n: 10
Hasil penjumlahan: 55
● PS C:\Users\M S I> 
```

## Deskripsi Program

Program di atas menggunakan fungsi rekursif untuk menghitung jumlah bilangan dari 1 hingga nilai tertentu yang dimasukkan oleh pengguna, yaitu `n`. Fungsi `jumlahRekursif` menerima satu parameter `n` dan memeriksa kondisi dasar, di mana jika `n` bernilai 1, maka fungsi mengembalikan nilai 1 dan menghentikan rekursi. Kondisi dasar ini sangat penting agar rekursi berhenti dan tidak berulang tanpa henti. Jika `n` tidak sama dengan 1, maka fungsi akan mengembalikan hasil penjumlahan `n` dengan hasil dari pemanggilan fungsi `jumlahRekursif` untuk nilai `n-1`. Dengan pendekatan ini, `jumlahRekursif` secara bertahap menambah setiap bilangan hingga mencapai nilai `n`, yang kemudian diakumulasi dari 1 hingga `n` melalui panggilan rekursif bertingkat.

Dalam fungsi `main`, program meminta masukan dari pengguna berupa nilai integer `n`, yang menandakan batas akhir dari penjumlahan yang ingin dilakukan. Setelah mendapatkan input, program memanggil `jumlahRekursif` dengan parameter `n`, yang kemudian menghitung total penjumlahan dari 1 hingga `n` dan mengembalikannya ke `main`. Hasilnya dicetak dalam format yang menunjukkan hasil akhir dari penjumlahan tersebut. Penggunaan rekursi pada program ini menjadikan proses penjumlahan lebih intuitif dan ringkas tanpa perlu menggunakan perulangan eksplisit seperti `for` atau `while`, karena setiap nilai dihitung bertahap melalui pemanggilan berulang sampai kondisi dasar tercapai.

### Guided 1.3

#### Sourcecode

```
// 2311102266_Hanif reyhan zharan adrytona

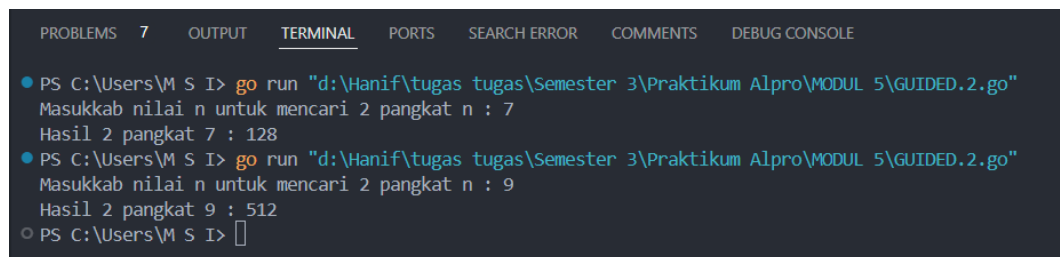
package main

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))
}
```

#### Screenshoot Output



```
PROBLEMS 7 OUTPUT TERMINAL PORTS SEARCH ERROR COMMENTS DEBUG CONSOLE

● PS C:\Users\M S I> go run "d:\Hanif\tugas tugas\Semester 3\Praktikum Alpro\MODUL 5\GUIDED.2.go"
Masukkan nilai n untuk mencari 2 pangkat n : 7
Hasil 2 pangkat 7 : 128
● PS C:\Users\M S I> go run "d:\Hanif\tugas tugas\Semester 3\Praktikum Alpro\MODUL 5\GUIDED.2.go"
Masukkan nilai n untuk mencari 2 pangkat n : 9
Hasil 2 pangkat 9 : 512
○ PS C:\Users\M S I> 
```

#### Deskripsi Program

Program di atas bertujuan untuk menghitung nilai pangkat dua dari suatu bilangan bulat `n` menggunakan fungsi rekursif. Fungsi `pangkatDua` menerima satu parameter integer `n` dan mengembalikan hasil dari 2 pangkat `n`. Fungsi ini dimulai dengan memeriksa kondisi dasar di mana, jika `n` bernilai 0, fungsi akan mengembalikan nilai 1 karena setiap bilangan berpangkat nol adalah 1. Jika `n` lebih dari 0, maka fungsi akan mengalikan 2 dengan hasil dari `pangkatDua(n-1)`. Proses ini dilakukan berulang kali sampai nilai `n` mencapai 0, di mana seluruh hasil perkalian akan diakumulasikan dan dihitung dari tahap akhir hingga awal.

Pada bagian `main`, program meminta masukan nilai `n` dari pengguna untuk menghitung hasil  $2^n$ . Setelah mendapatkan input, program memanggil fungsi `pangkatDua` dengan parameter `n` untuk memulai proses rekursi. Setelah fungsi `pangkatDua` selesai menghitung, hasilnya dikembalikan dan dicetak dalam format "Hasil 2 pangkat n :". Program ini menggunakan rekursi untuk menghitung pangkat dua, sehingga proses perhitungannya lebih sederhana dan tidak memerlukan perulangan eksplisit. Pendekatan ini efektif karena memanfaatkan pemanggilan fungsi bertingkat untuk melakukan operasi yang berulang, yakni mengalikan angka 2 sebanyak `n` kali secara bertahap.

#### Guided 1.4

#### Sourcecode

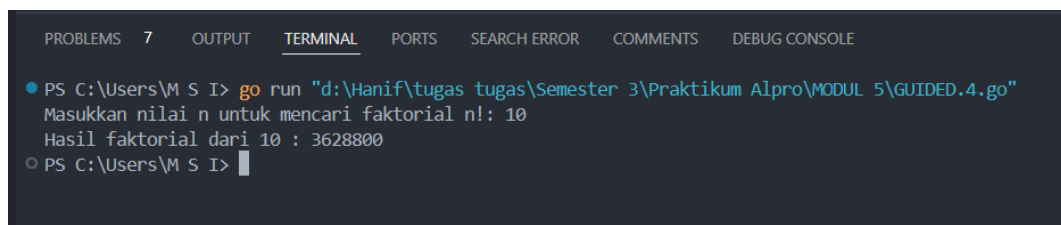
```
package main

import "fmt"

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

func main() {
    var n int
    fmt.Print("Masukkan nilai n untuk mencari faktorial n!: ")
    fmt.Scanln(&n)
    fmt.Println("Hasil faktorial dari", n, ":",
        faktorial(n))
}
```

#### Screenshoot Output



```
PROBLEMS 7 OUTPUT TERMINAL PORTS SEARCH ERROR COMMENTS DEBUG CONSOLE

● PS C:\Users\M S I> go run "d:\Hanif\tugas tugas\Semester 3\Praktikum Alpro\MODUL 5\GUIDED.4.go"
Masukkan nilai n untuk mencari faktorial n!: 10
Hasil faktorial dari 10 : 3628800
○ PS C:\Users\M S I> █
```

#### Deskripsi Program

Program di atas bertujuan untuk menghitung nilai faktorial dari sebuah bilangan bulat positif `n` menggunakan fungsi rekursif. Fungsi `faktorial` menerima satu parameter integer `n` dan mengembalikan hasil faktorial dari `n`. Faktorial sendiri merupakan hasil perkalian dari suatu bilangan dengan seluruh bilangan bulat positif di bawahnya hingga 1. Dalam fungsi `faktorial`, terdapat dua kondisi: jika `n` adalah 0 atau 1, maka fungsi mengembalikan nilai 1, karena `0!` dan `1!` bernilai 1 secara definisi. Jika `n` lebih besar dari 1, fungsi akan memanggil dirinya sendiri (`faktorial(n-1)`) dan mengalikan `n` dengan hasil dari pemanggilan tersebut. Proses ini dilakukan berulang kali hingga `n` mencapai nilai 1, di mana hasil perkalian bertahap akan diakumulasikan untuk menghasilkan nilai faktorial.

Pada fungsi `main`, program meminta input dari pengguna untuk menentukan nilai `n` yang faktorialnya ingin dihitung. Setelah pengguna memasukkan nilai `n`, program memanggil fungsi `faktorial` dengan `n` sebagai parameter. Hasil perhitungan kemudian dicetak dalam format "Hasil faktorial dari n: ". Program ini menggunakan rekursi untuk menghitung faktorial, yang membuatnya efektif dan ringkas, terutama karena rekursi memecah masalah menjadi masalah yang lebih kecil secara bertahap. Dengan pendekatan ini, tidak perlu menggunakan perulangan eksplisit, karena pemanggilan fungsi berulang kali secara otomatis menangani proses perkalian bertahap.



### III. UNGUIDED

#### Soal Studi Case

Deret fibonacci adalah sebuah deret dengan nilai suku ke-0 dan ke-1 adalah 0 dan 1, dan nilai suku ke-n selanjutnya adalah hasil penjumlahan dua suku sebelumnya. Secara umum dapat diformulasikan  $S_2 = S_{n-1} + S_{n-2}$ . Berikut ini adalah contoh nilai deret fibonacci hingga suku ke-10. Buatlah program yang mengimplementasikan fungsi rekursif pada deret fibonacci tersebut.

#### Sourcecode

```
// 2311102266_Hanif reyhan zharan adrytona

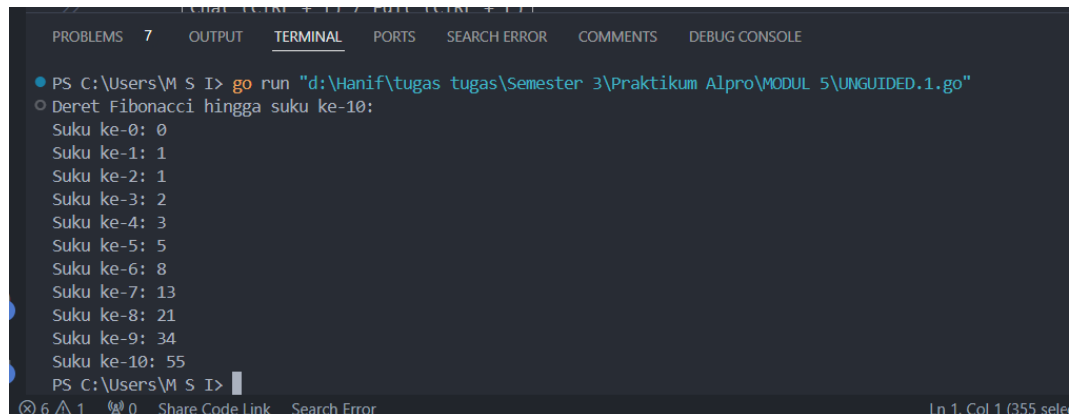
package main

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))
    }
}
```

#### Screenshoot Output

A screenshot of a Go IDE terminal window. The terminal shows the command `go run "d:\Hanif\tugas tugas\Semester 3\Praktikum Alpro\MODUL 5\UNGUIDED.1.go"` being executed. The output displays the Fibonacci sequence up to the 10th term: "Deret Fibonacci hingga suku ke-10:", followed by "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", and "Suku ke-10: 55". The terminal interface includes tabs for PROBLEMS, OUTPUT, TERMINAL, PORTS, SEARCH ERROR, COMMENTS, and DEBUG CONSOLE. The status bar at the bottom shows "Ln 1, Col 1 (355 sele...".

```
PS C:\Users\M S I> go run "d:\Hanif\tugas tugas\Semester 3\Praktikum Alpro\MODUL 5\UNGUIDED.1.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 C:\Users\M S I>
```

## Deskripsi Program

Program di atas bertujuan untuk menampilkan deret Fibonacci hingga suku ke-`n` menggunakan pendekatan rekursif. Fungsi `fibonacci` mendefinisikan perhitungan rekursif untuk menemukan nilai suku ke-`n` dalam deret Fibonacci, di mana nilai dari suku pertama (`fibonacci(0)`) adalah 0 dan suku kedua (`fibonacci(1)`) adalah 1. Fungsi ini menggunakan kondisi dasar untuk mengembalikan `n` saat `n` bernilai 0 atau 1, dan jika tidak, fungsi tersebut memanggil dirinya sendiri untuk menghitung jumlah dua suku sebelumnya (`fibonacci(n-1) + fibonacci(n-2)`). Dengan pendekatan ini, nilai setiap suku dalam deret Fibonacci dapat dihitung secara bertahap dengan menjumlahkan dua nilai sebelumnya dalam deret tersebut.

Pada fungsi `main`, program mendefinisikan variabel `n` dengan nilai 10, yang berarti deret Fibonacci akan dicetak hingga suku ke-10. Melalui perulangan `for`, program memanggil fungsi `fibonacci` untuk menghitung setiap suku dari 0 hingga `n`, dan hasilnya dicetak satu per satu. Setiap nilai suku dicetak dalam format "Suku ke-i: hasil". Program ini memberikan gambaran mengenai bagaimana deret Fibonacci dibangun dari nilai-nilai sebelumnya secara rekursif, meskipun pendekatan rekursif ini tidak terlalu efisien untuk nilai `n` yang besar karena banyaknya perhitungan berulang yang terjadi.

## Unguided 2

SOAL :

Buatlah sebuah program yang digunakan untuk menampilkan pola bintang berikut ini dengan menggunakan fungsi rekursif. N adalah masukan dari user.

## Sourcecode

```
// 2311102266_Hanif reyhan zharan adrytona

package main

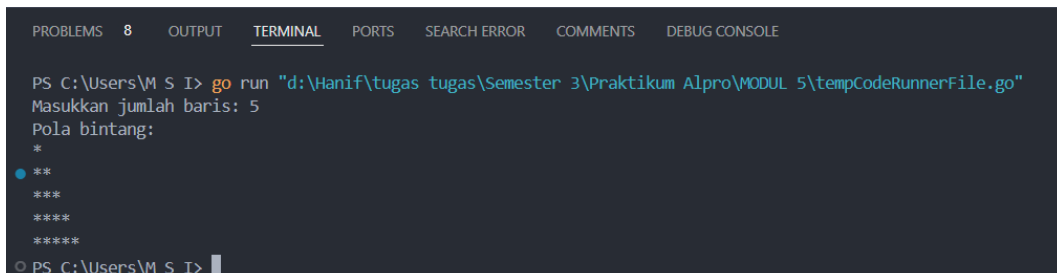
import "fmt"

func printStars(n int) {
    if n > 0 {
        fmt.Print("*")
        printStars(n - 1)
    }
}

func printPattern(n int) {
    if n > 0 {
        printPattern(n - 1)
        printStars(n)
        fmt.Println()
    }
}

func main() {
    var n int
    fmt.Print("Masukkan jumlah baris: ")
    fmt.Scan(&n)
    fmt.Println("Pola bintang:")
    printPattern(n)
}
```

## Screenshoot Output



```
PROBLEMS 8 OUTPUT TERMINAL PORTS SEARCH ERROR COMMENTS DEBUG CONSOLE

PS C:\Users\M S I> go run "d:\Hanif\tugas tugas\Semester 3\Praktikum Alpro\MODUL 5\tempCodeRunnerFile.go"
Masukkan jumlah baris: 5
Pola bintang:
*
**
***
****
*****

PS C:\Users\M S I>
```

## Deskripsi Program

Program di atas menghasilkan pola bintang berbentuk segitiga bertingkat berdasarkan jumlah baris yang dimasukkan pengguna. Fungsi utama dalam program ini adalah `printPattern`, yang mencetak baris bintang dari satu bintang hingga mencapai jumlah bintang yang sama dengan nilai input `n`. Pada setiap

pemanggilan `printPattern`, program menggunakan rekursi untuk mencetak bintang dari baris terkecil hingga baris yang diinginkan. Setiap pemanggilan `printPattern` juga memanggil `printStars`, sebuah fungsi rekursif lain yang bertugas mencetak jumlah bintang pada baris tertentu.

Fungsi `printStars` menggunakan parameter `n` sebagai pengatur jumlah bintang yang dicetak pada setiap baris. Jika `n` lebih dari nol, fungsi ini akan mencetak satu bintang dan memanggil dirinya sendiri, mengurangi nilai `n` satu per satu hingga mencapai nol. Fungsi `printPattern` secara efektif mengatur jumlah baris, sedangkan `printStars` mengatur jumlah bintang di setiap baris. Gabungan kedua fungsi ini menghasilkan pola segitiga bintang yang simetris, yang menambah baris satu per satu, memberikan hasil akhir pola bintang sesuai input dari pengguna.

### UNGUIDED 3

Buatlah program yang mengimplementasikan rekursif untuk menampilkan faktor bilangan dari suatu N, atau bilangan yang apa saja yang habis membagi N. Masukan terdiri dari sebuah bilangan bulat positif N. Keluaran terdiri dari barisan bilangan yang menjadi faktor dari N (terurut dari 1 hingga N ya).

### Sourcecode

```
// 2311102266_Hanif reyhan zharan adrytona

package main

import "fmt"

func faktor(n, bagi int) {
    if bagi > n {
        return
    }
    if n%bagi == 0 {
        fmt.Printf("%d ", bagi)
    }
    faktor(n, bagi+1)
}

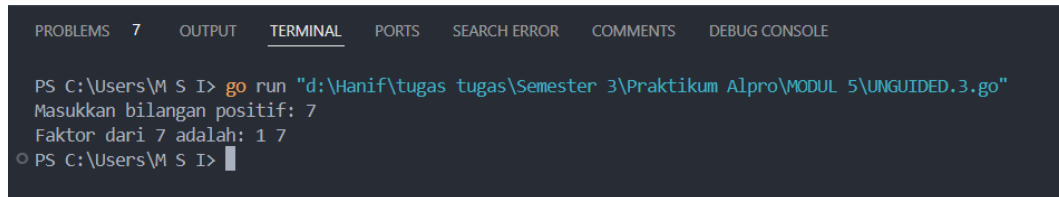
func main() {
    var n int
    fmt.Print("Masukkan bilangan positif: ")
    fmt.Scan(&n)
```

```

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

```

## Screenshoot Output



```

PROBLEMS 7 OUTPUT TERMINAL PORTS SEARCH ERROR COMMENTS DEBUG CONSOLE

PS C:\Users\M S I> go run "d:\Hanif\tugas tugas\Semester 3\Praktikum Alpro\MODUL 5\UNGUIDED.3.go"
Masukkan bilangan positif: 7
Faktor dari 7 adalah: 1 7
PS C:\Users\M S I>

```

## Deskripsi Program

Program di atas adalah program Go yang berfungsi untuk mencari dan mencetak semua faktor dari bilangan positif `n` yang dimasukkan oleh pengguna. Program ini menggunakan rekursi dalam fungsi `faktor` untuk memeriksa setiap bilangan dari 1 hingga `n`, memverifikasi apakah bilangan tersebut merupakan faktor dari `n`. Pada fungsi `faktor`, parameter `bagi` akan bertambah satu pada setiap pemanggilan rekursif untuk memeriksa setiap bilangan hingga nilai `bagi` melebihi `n`, yang berarti bahwa proses pencarian faktor telah selesai.

Fungsi `faktor` bekerja dengan mengecek apakah `n % bagi == 0`, yang menunjukkan bahwa `bagi` adalah faktor dari `n`. Jika benar, nilai `bagi` akan dicetak sebagai faktor dari `n`. Proses ini diulang terus dengan nilai `bagi` yang bertambah 1 setiap kali fungsi `faktor` dipanggil ulang secara rekursif. Setelah seluruh bilangan dari 1 hingga `n` telah diperiksa, program kembali ke fungsi `main` dan mencetak daftar faktor `n`. Program ini menunjukkan cara memanfaatkan rekursi untuk menyelesaikan masalah iteratif secara efektif dalam bahasa Go.

## UNGUIDED 4

Buatlah program yang menimplementasikan rekursif untuk menampilkan barisan bilangan tertentu. Masukan terdiri dari sebuah bilangan bulat positif N. Keluaran terdiri dari barisan bilangan dari N hingga 1 dan kembali ke N.

## Sourcecode

```

// 2311102266_Hanif reyhan zharan adrytona

package main

```

```

import (
    "fmt"
)

func printSequence(n, current int) {
    if current < 1 {
        return
    }
    fmt.Print(current, " ")
    if current > 1 {
        printSequence(n, current-1)
    }
    fmt.Print(current, " ")
}

func main() {
    var n int
    fmt.Print("Masukkan bilangan : ")
    fmt.Scan(&n)
    printSequence(n, n)
    fmt.Println()
}

```

### Screenshoot Output

```

PROBLEMS 7 OUTPUT TERMINAL PORTS SEARCH ERROR COMMENTS DEBUG CONSOLE

PS C:\Users\M S I> go run "d:\Hanif\tugas tugas\Semester 3\Praktikum Alpro\MODUL 5\UNGUIDED.4.go"
Masukkan bilangan : 8
8 7 6 5 4 3 2 1 1 2 3 4 5 6 7 8
PS C:\Users\M S I>

```

### Deskripsi Program

Program di atas adalah implementasi dalam bahasa Go yang bertujuan untuk mencetak urutan angka berdasarkan bilangan bulat positif `n` yang dimasukkan oleh pengguna. Fungsi utama dari program ini adalah `printSequence`, yang mengambil dua parameter: `n`, yang merupakan bilangan positif yang dimasukkan, dan `current`, yang berfungsi sebagai penghitung untuk mencetak angka. Ketika fungsi `printSequence` dipanggil, ia pertama-tama memeriksa apakah `current` kurang dari 1. Jika ya, fungsi akan berhenti (base case). Jika tidak, fungsi akan mencetak nilai `current`, kemudian memanggil dirinya sendiri secara rekursif dengan nilai `current` yang dikurangi 1, sebelum mencetak nilai `current` lagi setelah pemanggilan rekursif selesai.

Sebagai contoh, jika pengguna memasukkan angka 5, output yang dihasilkan akan menjadi: "5 4 3 2 1 1 2 3 4 5". Ini terjadi karena fungsi mencetak angka dari `current` hingga 1 dalam urutan menurun, kemudian setelah mencapai 1, fungsi kembali mencetak angka dalam urutan menaik, menciptakan simetri dalam output. Program ini menunjukkan penggunaan rekursi untuk menghasilkan dua bagian dari urutan yang saling mencerminkan, serta cara untuk mencetak hasil dengan mudah di dalam bahasa Go.

## UNGUIDED 5

Buatlah program yang menimplementasikan rekursif untuk menampilkan barisan bilangan ganjil. Masukan terdiri dari sebuah bilangan bulat positif N. Keluaran terdiri dari barisan bilangan ganjil dari 1 hingga N.

## Sourcecode

```
// 2311102266_Hanif reyhan zharan adrytona

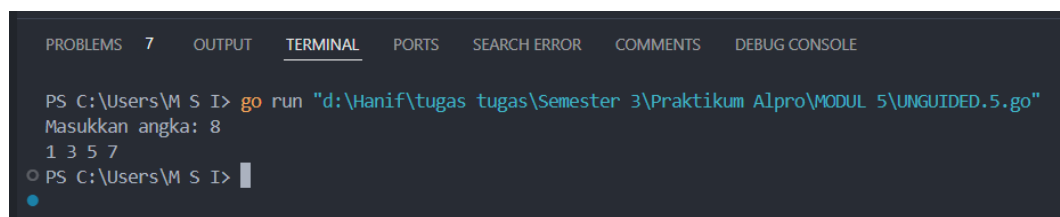
package main

import (
    "fmt"
)

func ganjil(n, current int) {
    if current <= n {
        fmt.Print(current, " ")
        ganjil(n, current+2)
    }
}

func main() {
    var n int
    fmt.Print("Masukkan angka: ")
    fmt.Scan(&n)
    ganjil(n, 1)
    fmt.Println()
}
```

## Screenshoot Output



```
PROBLEMS 7 OUTPUT TERMINAL PORTS SEARCH ERROR COMMENTS DEBUG CONSOLE

PS C:\Users\M S I> go run "d:\Hanif\tugas tugas\Semester 3\Praktikum Alpro\MODUL 5\UNGUIDED.5.go"
Masukkan angka: 8
1 3 5 7
PS C:\Users\M S I>
```

## Deskripsi Program

Program di atas ditulis dalam bahasa Go dan berfungsi untuk mencetak semua bilangan ganjil mulai dari 1 hingga bilangan bulat positif `n` yang dimasukkan oleh pengguna. Fungsi utama dalam program ini adalah `ganjil`, yang menerima dua parameter: `n`, batas atas yang ditentukan oleh pengguna, dan `current`, yang digunakan untuk melacak bilangan ganjil yang sedang dicetak. Pada setiap pemanggilan, fungsi memeriksa apakah `current` kurang dari atau sama dengan `n`. Jika iya, ia akan mencetak nilai `current` dan kemudian memanggil dirinya sendiri secara rekursif dengan menambahkan 2 pada `current`, yang menjamin bahwa hanya bilangan ganjil yang akan dicetak.

Sebagai contoh, jika pengguna memasukkan angka 9, output yang dihasilkan akan menjadi "1 3 5 7 9". Program ini menggunakan pendekatan rekursif untuk menghasilkan dan mencetak urutan bilangan ganjil, mulai dari 1 dan bertambah dua setiap kali pemanggilan fungsi dilakukan. Dengan cara ini, program tidak hanya menampilkan bilangan ganjil tetapi juga menunjukkan penerapan konsep rekursi dalam pemrograman, di mana fungsi dapat memanggil dirinya sendiri untuk menyelesaikan tugas berulang hingga kondisi tertentu terpenuhi.

## UNGUIDED 6

Buatlah program yang mengimplementasikan rekursif untuk menampilkan barisan bilangan ganjil. Masukan terdiri dari sebuah bilangan bulat positif N. Keluaran terdiri dari barisan bilangan ganjil dari 1 hingga N.

## Sourcecode

```
// 2311102266_Hanif reyhan zharan adrytona

package main

import "fmt"

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

func main() {
    var x, y int
```

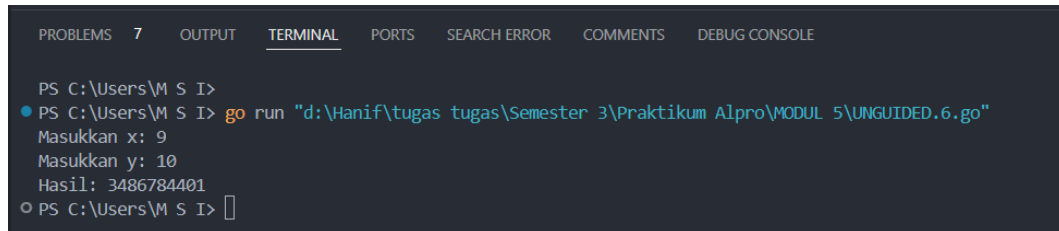


```

    fmt.Print("Masukkan x: ")
    fmt.Scanln(&x)
    fmt.Print("Masukkan y: ")
    fmt.Scanln(&y)
    fmt.Println("Hasil:", pangkat(x, y))
}

```

## Screenshoot Output



```

PROBLEMS 7 OUTPUT TERMINAL PORTS SEARCH ERROR COMMENTS DEBUG CONSOLE

PS C:\Users\M S I>
● PS C:\Users\M S I> go run "d:\Hanif\tugas tugas\Semester 3\Praktikum Alpro\MODUL 5\UNGUIDED.6.go"
Masukkan x: 9
Masukkan y: 10
Hasil: 3486784401
○ PS C:\Users\M S I>

```

## Deskripsi Program

Program di atas merupakan implementasi dalam bahasa Go untuk menghitung hasil pangkat dari bilangan bulat `x` yang dipangkatkan dengan bilangan bulat `y` menggunakan pendekatan rekursif. Fungsi `pangkat` di dalam program ini menerima dua parameter, yaitu `x` sebagai basis dan `y` sebagai eksponen. Program ini mengimplementasikan tiga kondisi dalam fungsi rekursif: jika `y` sama dengan 0, hasilnya adalah 1 (karena setiap bilangan pangkat 0 sama dengan 1), jika `y` sama dengan 1, hasilnya adalah `x` (karena setiap bilangan pangkat 1 sama dengan dirinya sendiri), dan untuk nilai `y` lebih dari 1, fungsi akan memanggil dirinya sendiri dengan menurunkan nilai `y` hingga mencapai kondisi dasar.

Pada bagian `main`, program meminta input dari pengguna untuk dua bilangan, yaitu `x` dan `y`. Setelah menerima input, program memanggil fungsi `pangkat` dengan parameter `x` dan `y`, dan mencetak hasilnya ke layar. Program ini menunjukkan bagaimana rekursi dapat digunakan untuk menyelesaikan masalah yang dapat dibagi menjadi sub-masalah yang lebih kecil, serta memberikan pemahaman yang baik tentang bagaimana pangkat bilangan dapat dihitung tanpa menggunakan operasi perkalian langsung di luar definisi rekursif.