LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL VI

Rekursif

Disusun Oleh :

Nadhif Atha Zaki / 2311102007

IF-11-05

Dosen Pengampu :

Arif Amrulloh, S.Kom., M.Kom

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Pengulangan rekursif adalah teknik dalam pemrograman di mana suatu fungsi memanggil dirinya sendiri untuk menyelesaikan tugas yang lebih kecil hingga mencapai kondisi dasar yang menghentikan proses tersebut. Ini merupakan salah satu pendekatan yang sering digunakan dalam pemrograman fungsional dan algoritma. Berikut adalah beberapa dasar teori mengenai pengulangan rekursif:

1. Dasar Rekursi: Setiap fungsi rekursif harus memiliki satu atau lebih kondisi dasar (base case) yang menentukan kapan fungsi tidak lagi memanggil dirinya sendiri. Tanpa kondisi dasar, rekursi akan berlanjut tanpa akhir, menyebabkan stack overflow.
2. Pengurangan Masalah: Fungsi rekursif sering kali menyelesaikan masalah dengan membagi masalah besar menjadi masalah yang lebih kecil. Misalnya, untuk menghitung faktorial dari suatu bilangan, faktorial nnn (ditulis n!n!n!) dapat dihitung sebagai n×(n−1)!n \times (n-1)!n×(n−1)!.
3. Struktur Rekursif: Rekursi dapat dibedakan menjadi dua jenis:
    - Rekursi langsung: Fungsi memanggil dirinya sendiri secara langsung.
    - Rekursi tidak langsung: Fungsi memanggil fungsi lain yang pada gilirannya memanggil fungsi awal.
4. Penggunaan Memori: Rekursi menggunakan stack untuk menyimpan informasi tentang pemanggilan fungsi. Setiap kali fungsi dipanggil, informasi baru ditambahkan ke stack, dan ketika fungsi selesai, informasi tersebut dihapus. Ini dapat menyebabkan penggunaan memori yang lebih tinggi dibandingkan dengan iterasi, terutama jika kedalaman rekursi sangat besar.
5. Contoh Penggunaan: Rekursi sering digunakan dalam algoritma pemrograman seperti pencarian (searching), pengurutan (sorting), pemrograman dinamis (dynamic programming), dan masalah yang berkaitan dengan struktur data seperti pohon (tree) dan grafik (graph).

Pengulangan rekursif memberikan cara yang elegan untuk menyelesaikan banyak masalah, tetapi penting untuk memastikan bahwa solusi yang dihasilkan efisien dan tidak menyebabkan penggunaan sumber daya yang berlebihan.

II. GUIDED

1.

Membuat baris bilangan dari n hingga 1

Base-case: bilangan == 1

Sourcecode

```
package main

import "fmt"

func main() {
    var n int
    fmt.Scan(&n)
    baris(n)
}
func baris(bilangan int) {
    if bilangan == 1 {
        fmt.Println(1)
    } else {
        fmt.Println(bilangan)
        baris(bilangan - 1)
    }
}
```

Screenshoot Output

```
PS D:\Nathhh\matkul\smst 3\Praktikum Alpro 2\Modul-VI> go run "d:
8
8
7
6
5
4
3
2
1
PS D:\Nathhh\matkul\smst 3\Praktikum Alpro 2\Modul-VI>
```

Deskripsi Program

Program ini membaca sebuah bilangan `n` dari input pengguna dan menampilkan bilangan tersebut hingga angka 1 dalam urutan menurun. Fungsi utama `main` menerima input `n` dan kemudian memanggil fungsi `baris` untuk mencetak urutan angka. Fungsi `baris` adalah fungsi rekursif yang menampilkan nilai `bilangan` saat ini, lalu memanggil dirinya sendiri dengan nilai `bilangan` yang dikurangi 1, sehingga mencetak urutan menurun. Basis rekursinya adalah ketika `bilangan` mencapai 1, di mana fungsi hanya mencetak "1" dan berhenti. Hasil akhirnya adalah urutan angka dari `n` sampai 1, masing-masing pada baris baru.

2.

Menghitung hasil penjumlahan 1 hingga n

Base-case: n == 1

Sourcecode

```go
package main

import "fmt"

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

Screenshoot Output

```
PS D:\Nathhh\matkul\smst 3\Praktikum Alpro 2\Modul-VI> go run
7
28
PS D:\Nathhh\matkul\smst 3\Praktikum Alpro 2\Modul-VI>
```

Deskripsi Program

Program ini menampilkan pola bintang bertingkat sesuai dengan jumlah baris yang diminta pada setiap kasus uji. Fungsi `printStars` menggunakan rekursi untuk mencetak pola bintang, di mana setiap baris berisi jumlah bintang yang meningkat sesuai dengan urutan barisnya. Basis rekursi berhenti jika nilai `current` melebihi batas `n`, yaitu jumlah baris yang diinginkan. Di fungsi `main`, pengguna diminta memasukkan jumlah kasus uji, kemudian memasukkan nilai `n` untuk setiap kasus, yang menentukan jumlah baris bintang yang akan dicetak. Program akhirnya menampilkan hasil setiap kasus dalam format tabel yang menunjukkan nomor kasus, input yang diberikan, dan pola bintang hasilnya.

3.

Mencari dua pangkat n atau 2^n

Base-case: n == 0

Sourcecode

```go
package main

import "fmt"

func pangkat(n int) int {
    if n == 0 {
        return 1
    } else {
        return 2 * pangkat(n-1)
    }
}
func main() {
    var n int
    fmt.Print("MasukKan nilai n: ")
    fmt.Scan(&n)
    fmt.Println("Hasil dari 2 pangkat", n,
        "adalah:", pangkat(n))
}
```

Screenshoot Output

```
PS D:\Nathhh\matkul\smst 3\Praktikum Alpro 2\Modul-VI> go r
MasukKan nilai n: 10
Hasil dari 2 pangkat 10 adalah: 1024
PS D:\Nathhh\matkul\smst 3\Praktikum Alpro 2\Modul-VI>
```

Deskripsi Program

Program ini menghitung nilai dari 2 pangkat `n` menggunakan fungsi rekursif. Fungsi `pangkat` menerima parameter `n`, yang merupakan eksponen, dan mengembalikan hasil dari 2 pangkat `n`. Basis rekursinya adalah ketika `n` sama dengan 0, di mana fungsi akan mengembalikan nilai 1 (karena $2^0 = 1$). Untuk nilai `n` lebih besar dari 0, fungsi akan mengalikan 2 dengan hasil pemanggilan fungsi `pangkat` dengan eksponen `n-1`, sehingga menghasilkan nilai pangkat yang diinginkan. Di fungsi `main`, pengguna diminta memasukkan nilai `n`, lalu program akan menampilkan hasil dari 2 pangkat `n`.

4.

Mencari nilai faktorial atau n!

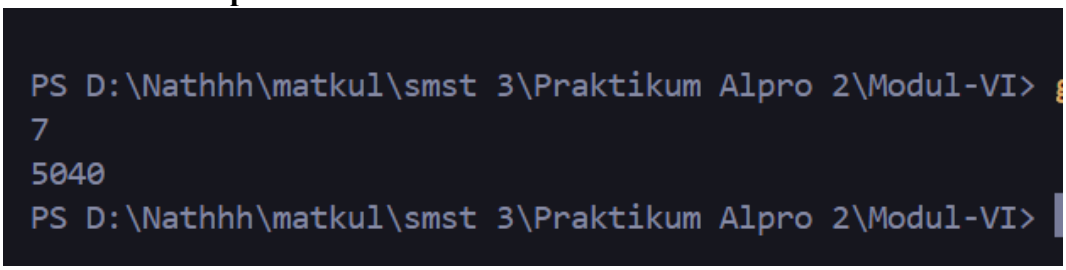
Base-case: n == 0 atau n == 1

Sourcecode

```go
package main

import "fmt"

var n int

func faktorial(n int) int {
    if n == 0 || n == 1 {
        return 1
    } else {
        return n * faktorial(n-1)
    }
}
func main() {
    fmt.Scan(&n)
    fmt.Println(faktorial(n))
}
```

```

```

Screenshoot Output



Deskripsi Program

Program ini menghitung nilai faktorial dari bilangan bulat `n` yang dimasukkan oleh pengguna. Fungsi `faktorial` adalah fungsi rekursif yang menerima parameter `n`, dan mengembalikan 1 jika `n` sama dengan 0 atau 1 (karena faktorial dari 0 dan 1 adalah 1). Jika `n` lebih besar dari 1, fungsi akan mengalikan `n` dengan hasil pemanggilan dirinya sendiri dengan parameter `n-1`, sehingga menghasilkan faktorial dari `n`. Di dalam fungsi `main`, program membaca input `n` dari pengguna dan kemudian mencetak hasil faktorial yang dihitung dengan memanggil fungsi `faktorial`. Hasil akhirnya adalah output nilai faktorial dari bilangan yang dimasukkan.

III. UNGUIDED

1.

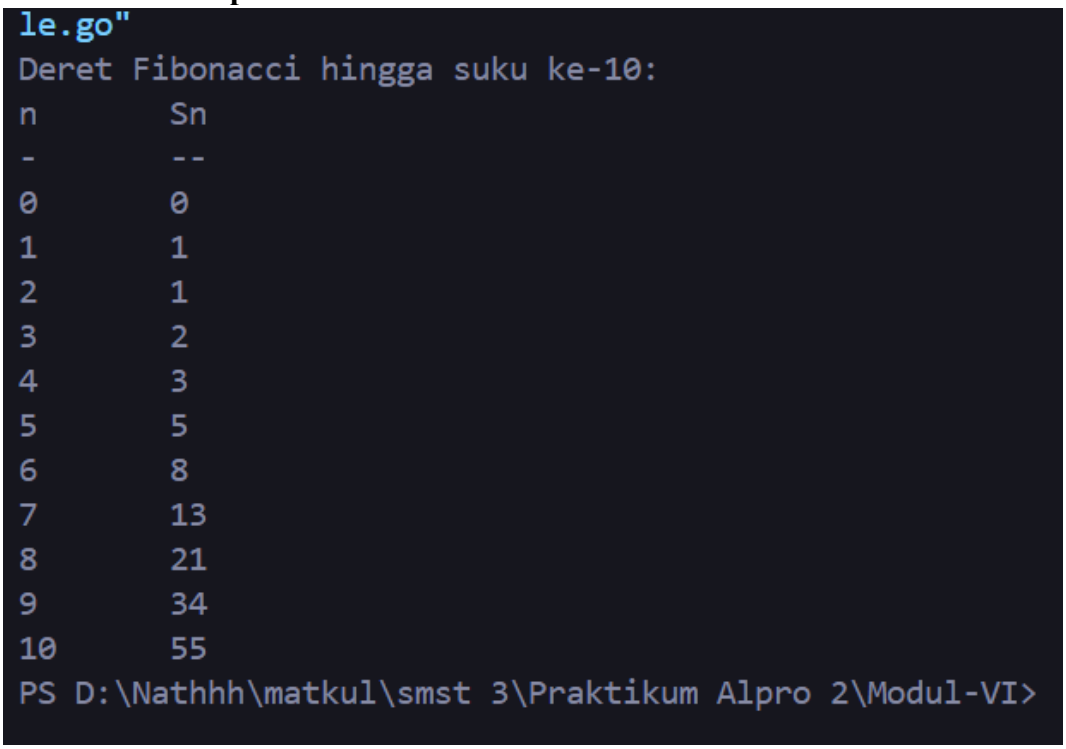
Deret fibonacci adalah sebuah deret dengan nilai suku ke-0 dan ke-1 adalah 0 dan 1, dan nilai suku ke-n selanjutnya adalah hasil penjumlahan dua suku sebelumnya. Secara umum dapat diformulasikan $S_n = S_{n-1} + S_{n-2}$. Berikut ini adalah contoh nilai deret fibonacci hingga suku ke-10. Buatlah program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

n	0	1	2	3	4	5	6	7	8	9	10
S_n	0	1	1	2	3	5	8	13	21	34	55

Sourcecode

```go
package main

import "fmt"

// Fungsi fibonacci menghitung nilai suku ke-n dari deret Fibonacci
func fibonacci(n int) int {
    // Basis rekursi: jika n kurang dari atau sama dengan 1, kembalikan nilai n
    if n <= 1 {
        return n
    }
    // Rekursi: hitung nilai Sn sebagai penjumlahan dari dua suku sebelumnya (Sn-1 + Sn-2)
    return fibonacci(n-1) + fibonacci(n-2)
}

func main() {
    // Menampilkan header untuk deret Fibonacci hingga suku ke-10
    fmt.Println("Deret Fibonacci hingga suku ke-10:")
    fmt.Println("n\tSn")
    fmt.Println("-\t--")

    // Loop untuk menghitung dan menampilkan setiap suku dari 0 hingga 10
    for i := 0; i <= 10; i++ {
        fmt.Printf("%d\t%d\n", i, fibonacci(i))
    }
}
```

Screenshoot Output



Deskripsi Program

Program ini digunakan untuk menghitung dan menampilkan deret Fibonacci hingga suku ke-10 menggunakan metode rekursi. Fungsi `fibonacci` menerima parameter `n` dan mengembalikan nilai suku ke-n dari deret Fibonacci. Jika nilai `n` kurang dari atau sama dengan 1, fungsi ini mengembalikan `n` sebagai basis rekursi. Jika tidak, fungsi ini menghitung suku Fibonacci dengan menjumlahkan hasil dari pemanggilan dirinya sendiri untuk dua suku sebelumnya (Sn-1 dan Sn-2). Di dalam fungsi `main`, program mencetak header untuk deret Fibonacci dan menggunakan loop untuk menghitung dan menampilkan setiap suku dari indeks 0 hingga 10, beserta nilai suku tersebut.

2.

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

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	5	* ** *** **** *****
2	1	*
3	3	* ** ***

Sourcecode

```go
package main

import (
    "fmt"
)

// Fungsi rekursif untuk menampilkan pola bintang
func printStars(n int, current int) {
    // Basis rekursi: berhenti jika current lebih besar dari n
    if current > n {
        return
    }

    // Cetak bintang sebanyak nilai current
    for i := 0; i < current; i++ {
        fmt.Print("*")
    }
    fmt.Println()

    // Panggilan rekursif untuk mencetak baris berikutnya
    printStars(n, current+1)
}
```

```go
func main() {
    var jumlahTest int
    fmt.Print("Masukkan jumlah kasus uji: ")
    fmt.Scan(&jumlahTest)

    // Array untuk menyimpan setiap nilai input
    inputs := make([]int, jumlahTest)

    // Mengambil input untuk setiap kasus uji
    for i := 0; i < jumlahTest; i++ {
        fmt.Printf("Masukan #%d: ", i+1)
        fmt.Scan(&inputs[i])
    }

    // Memproses dan menampilkan hasil untuk setiap kasus
uji
    fmt.Println("\nKeluaran:")
    fmt.Printf("No\tMasukan\tKeluaran\n")
    for i := 0; i < jumlahTest; i++ {
        fmt.Printf("%d\t%d\n", i+1, inputs[i])
        printStars(inputs[i], 1)
        fmt.Println()
    }
}
```

Screenshoot Output

```
PS D:\Nathhh\matkul\smst 3\Praktikum Alpro 2\Modul-VI> go
Masukkan jumlah kasus uji: 3
Masukan #1: 5
Masukan #2: 4
Masukan #3: 3

Keluaran:
No      Masukan Keluaran
1       5
*
**
***
****
*****

2       4
*
**
***
****

3       3
*
**
***

PS D:\Nathhh\matkul\smst 3\Praktikum Alpro 2\Modul-VI>
```

Deskripsi Program

Program ini digunakan untuk mencetak pola bintang yang bertingkat berdasarkan input jumlah baris dari pengguna. Fungsi `printStars` adalah fungsi rekursif yang mencetak bintang sebanyak nilai `current` pada setiap baris, dan akan berhenti ketika `current` lebih besar dari `n`. Di dalam fungsi ini, loop digunakan untuk mencetak bintang, dan setelah mencetak satu baris, fungsi memanggil dirinya sendiri dengan parameter `current` yang ditambah 1 untuk mencetak baris berikutnya. Di fungsi `main`, program meminta pengguna untuk memasukkan jumlah kasus uji dan setiap nilai input yang menunjukkan jumlah baris yang akan dicetak. Hasilnya, program menampilkan tabel yang menunjukkan nomor kasus, nilai input, dan pola bintang yang sesuai untuk setiap kasus.

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).

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	5	1 5
2	12	1 2 3 4 6 12

Sourcecode

```go
package main

import (
    "fmt"
)

// Fungsi rekursif untuk menemukan dan mencetak faktor dari bilangan
func findFactors(n int, current int) {
    // Basis rekursi: berhenti jika current lebih besar dari n
    if current > n {
        return
    }

    // Jika current adalah faktor dari n, cetak nilai current
    if n%current == 0 {
        fmt.Printf("%d ", current)
    }

    // Panggilan rekursif untuk mengecek faktor berikutnya
    findFactors(n, current+1)
}

func main() {
    var jumlahTest int
    fmt.Print("Masukkan jumlah kasus uji: ")
    fmt.Scan(&jumlahTest)
```

```
    // Array untuk menyimpan nilai input
    inputs := make([]int, jumlahTest)

    // Mengambil input dari pengguna untuk setiap kasus uji
    for i := 0; i < jumlahTest; i++ {
        fmt.Printf("Masukan #%d: ", i+1)
        fmt.Scan(&inputs[i])
    }

    // Memproses dan menampilkan hasil untuk setiap kasus
uji
    fmt.Println("\nNo\tMasukan\tKeluaran")
    for i := 0; i < jumlahTest; i++ {
        fmt.Printf("%d\t%d\t", i+1, inputs[i])
        findFactors(inputs[i], 1)
        fmt.Println()
    }
}
```

Screenshoot Output

```
PS D:\Nathhh\matkul\smst 3\Praktikum Alpro 2\Modul - IV> go run "d:\Nathhh\mat
"
Masukkan nilai awal deret: 15
15 46 23 70 35 106 53 160 80 40 20 10 5 16 8 4 2 1
PS D:\Nathhh\matkul\smst 3\Praktikum Alpro 2\Modul - IV>
```

Deskripsi Program

Program ini dirancang untuk menemukan dan mencetak faktor dari bilangan bulat yang dimasukkan oleh pengguna. Fungsi `findFactors` merupakan fungsi rekursif yang menerima dua parameter: `n`, yang adalah bilangan yang akan dicari faktornya, dan `current`, yang dimulai dari 1 dan digunakan untuk memeriksa setiap bilangan apakah merupakan faktor dari `n`. Basis rekursi fungsi ini terjadi ketika `current` lebih besar dari `n`, di mana fungsi akan berhenti. Jika `current` adalah faktor dari `n` (artinya `n` dibagi `current` tanpa sisa), maka nilai `current` akan dicetak. Di fungsi

`main`, program meminta pengguna untuk memasukkan jumlah kasus uji, kemudian menerima input untuk setiap kasus dan memanggil `findFactors` untuk mencetak faktor-faktor dari setiap bilangan tersebut, disajikan dalam format tabel yang menunjukkan nomor kasus, nilai masukan, dan faktor yang ditemukan.

4.

Buatlah program yang mengimplementasikan 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.

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	5	5 4 3 2 1 2 3 4 5
2	9	9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9

Sourcecode

```go
package main

import (
    "fmt"
)

// Fungsi rekursif untuk mencetak angka menurun
func printDescending(n int, current int) {
    if current < 1 {
        return
    }
    fmt.Printf("%d ", current)
    printDescending(n, current-1)
}

// Fungsi rekursif untuk mencetak angka menaik
func printAscending(n int, current int) {
    if current > n {
        return
    }
    fmt.Printf("%d ", current)
    printAscending(n, current+1)
}
```

```go
package main

import (
    "fmt"
)

// Fungsi rekursif untuk mencetak angka menurun
func printDescending(n int, current int) {
    if current < 1 {
        return
    }
    fmt.Printf("%d ", current)
    printDescending(n, current-1)
}

// Fungsi rekursif untuk mencetak angka menaik
func printAscending(n int, current int) {
    if current > n {
        return
    }
    fmt.Printf("%d ", current)
    printAscending(n, current+1)
}

// Fungsi untuk mencetak pola lengkap dari angka menurun ke menaik
func printPattern(n int) {
    printDescending(n, n)
    printAscending(n, 2)
}

func main() {
    var jumlahTest int
    fmt.Print("Masukkan jumlah kasus uji: ")
    fmt.Scan(&jumlahTest)

    // Array untuk menyimpan nilai input
    inputs := make([]int, jumlahTest)

    // Mengambil input untuk setiap kasus uji
    for i := 0; i < jumlahTest; i++ {
        fmt.Printf("Masukan #%d: ", i+1)
        fmt.Scan(&inputs[i])
    }
```

```
    // Memproses dan menampilkan hasil untuk setiap kasus
uji
    fmt.Println("\nNo\tMasukan\tKeluaran")
    for i := 0; i < jumlahTest; i++ {
        fmt.Printf("%d\t%d\t", i+1, inputs[i])
        printPattern(inputs[i])
        fmt.Println()
    }
}
```

Screenshoot Output

```
PS D:\Nathhh\matkul\smst 3\Praktikum Alpro 2\Modul-VI> go rur
Masukkan jumlah kasus uji: 3
Masukan #1: 6
Masukan #2: 5
Masukan #3: 4

No      Masukan Keluaran
1       6       6 5 4 3 2 1 2 3 4 5 6
2       5       5 4 3 2 1 2 3 4 5
3       4       4 3 2 1 2 3 4
PS D:\Nathhh\matkul\smst 3\Praktikum Alpro 2\Modul-VI>
```

Deskripsi Program

Program ini digunakan untuk mencetak pola angka yang dimulai dari angka yang diberikan oleh pengguna dalam urutan menurun, diikuti dengan urutan menaik. Fungsi `printDescending` adalah fungsi rekursif yang mencetak angka dari nilai `current` hingga 1. Basis rekursi terjadi ketika `current` kurang dari 1, di mana fungsi berhenti. Fungsi `printAscending` juga rekursif, yang mencetak angka dari 1 hingga nilai `n`, dan berhenti ketika `current` lebih besar dari `n`. Fungsi `printPattern` menggabungkan kedua fungsi tersebut dengan memanggil `printDescending` untuk mencetak angka menurun dari `n` dan kemudian `printAscending` untuk mencetak angka menaik dari 2 hingga `n`. Di fungsi `main`, program meminta pengguna untuk memasukkan jumlah kasus uji, mengumpulkan input, dan

menampilkan pola angka yang sesuai untuk setiap kasus uji dalam format tabel yang menunjukkan nomor kasus, nilai masukan, dan pola angka yang dihasilkan.

5.

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.

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	5	1 3 5
2	20	1 3 5 7 9 11 13 15 17 19

Sourcecode

```go
package main

import (
    "fmt"
)

// Fungsi rekursif untuk mencetak bilangan ganjil
func printOddNumbers(n int, current int) {
    // Basis rekursi: berhenti jika current lebih besar dari n
    if current > n {
        return
    }

    // Jika current adalah bilangan ganjil, cetak nilai current
    if current%2 != 0 {
        fmt.Printf("%d ", current)
    }

    // Panggilan rekursif untuk memeriksa bilangan berikutnya
    printOddNumbers(n, current+1)
}

func main() {
    var jumlahTest int
```

```
    fmt.Print("Masukkan jumlah kasus uji: ")
    fmt.Scan(&jumlahTest)

    // Array untuk menyimpan setiap nilai input
    inputs := make([]int, jumlahTest)

    // Mengambil input dari pengguna untuk setiap kasus uji
    for i := 0; i < jumlahTest; i++ {
        fmt.Printf("Masukan #%d: ", i+1)
        fmt.Scan(&inputs[i])
    }

    // Memproses dan menampilkan hasil untuk setiap kasus
uji
    fmt.Println("\nNo\tMasukan\tKeluaran")
    for i := 0; i < jumlahTest; i++ {
        fmt.Printf("%d\t%d\t", i+1, inputs[i])
        printOddNumbers(inputs[i], 1) // Mulai dari 1 untuk
mencetak bilangan ganjil
        fmt.Println()                 // Pindah baris untuk
kasus uji berikutnya
    }
}
```

Screenshoot Output

```
PS D:\Nathhh\matkul\smst 3\Praktikum Alpro 2\Modul-VI> go run "d
Masukkan jumlah kasus uji: 2
Masukan #1: 5
Masukan #2: 20

No      Masukan Keluaran
1       5       1 3 5
2       20      1 3 5 7 9 11 13 15 17 19
PS D:\Nathhh\matkul\smst 3\Praktikum Alpro 2\Modul-VI>
```

Deskripsi Program

Program ini dirancang untuk mencetak bilangan ganjil dari 1 hingga nilai yang dimasukkan oleh pengguna. Fungsi `printOddNumbers` adalah fungsi rekursif yang menerima dua parameter: `n`, yang merupakan batas atas untuk pencetakan bilangan ganjil, dan `current`, yang dimulai dari 1 dan digunakan untuk memeriksa setiap bilangan apakah merupakan bilangan ganjil. Basis rekursi terjadi ketika `current` lebih besar dari `n`, di mana fungsi akan berhenti. Dalam fungsi ini, jika `current` adalah bilangan ganjil (artinya sisa pembagian `current` dengan 2 tidak sama dengan 0), maka nilai `current` akan dicetak. Di dalam fungsi `main`, program meminta pengguna untuk memasukkan jumlah kasus uji, mengumpulkan input untuk setiap kasus, dan kemudian mencetak bilangan ganjil untuk setiap nilai masukan, disajikan dalam format tabel yang menunjukkan nomor kasus, nilai masukan, dan bilangan ganjil yang ditemukan.

6.

Buatlah program yang mengimplementasikan rekursif untuk mencari hasil pangkat dari dua buah bilangan.

Masukan terdiri dari bilangan bulat x dan y.

Keluaran terdiri dari hasil x dipangkatkan y.

Catatan: diperbolehkan menggunakan asterik "*", tapi dilarang menggunakan import "math".

Contoh masukan dan keluaran:

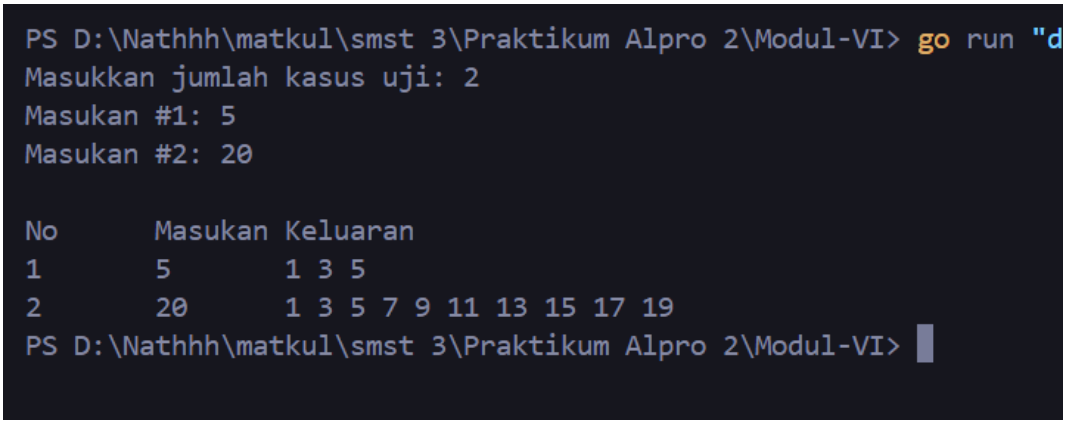
No	Masukan	Keluaran
1	2 2	4
2	5 3	125

Sourcecode

```go
package main

import (
    "fmt"
)

// Fungsi rekursif untuk menghitung nilai pangkat
func power(base int, exponent int) int {
    // Basis rekursi: jika eksponen adalah 0, kembalikan 1
    if exponent == 0 {
        return 1
    }
```

```
    // Hitung pangkat dengan mengalikan base dengan hasil
rekursif
    return base * power(base, exponent-1)
}

func main() {
    var jumlahTest int
    fmt.Print("Masukkan jumlah kasus uji: ")
    fmt.Scan(&jumlahTest)

    // Array untuk menyimpan nilai base dan eksponen setiap
input
    baseInputs := make([]int, jumlahTest)
    exponentInputs := make([]int, jumlahTest)

    // Mengambil input dari pengguna untuk setiap kasus uji
    for i := 0; i < jumlahTest; i++ {
        fmt.Printf("Masukan #%d (x y): ", i+1)
        fmt.Scan(&baseInputs[i], &exponentInputs[i])
    }

    // Memproses dan menampilkan hasil untuk setiap kasus
uji
    fmt.Println("\nNo\tMasukan\tKeluaran")
    for i := 0; i < jumlahTest; i++ {
        result := power(baseInputs[i], exponentInputs[i])
        fmt.Printf("%d\t%d %d\t%d\n",
            i+1,
            baseInputs[i],
            exponentInputs[i],
            result)
    }
}
```

Screenshoot Output



Deskripsi Program

Program ini bertujuan untuk menghitung nilai pangkat dari suatu bilangan berdasarkan input yang diberikan oleh pengguna. Fungsi `power` adalah fungsi rekursif yang mengambil dua parameter: `base` (bilangan yang akan dipangkatkan) dan `exponent` (pangkat yang akan diterapkan). Jika `exponent` bernilai 0, fungsi mengembalikan 1 sesuai dengan sifat matematika bahwa setiap bilangan pangkat nol adalah 1. Jika tidak, fungsi akan mengalikan `base` dengan hasil pangkat dari `base` yang dipangkatkan dengan `exponent - 1`, sehingga fungsi ini terus memanggil dirinya sendiri sampai mencapai basis rekursi. Dalam fungsi `main`, program meminta pengguna untuk memasukkan jumlah kasus uji, kemudian membaca pasangan nilai `base` dan `exponent` untuk setiap kasus, menghitung hasil pangkat menggunakan fungsi `power`, dan menampilkan hasilnya dalam format tabel yang menunjukkan nomor kasus, nilai masukan, dan hasil pangkat.