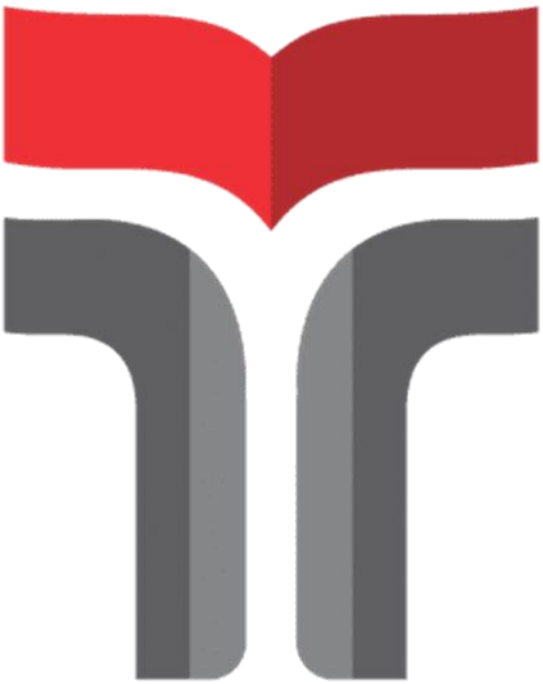
LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN 2

MODUL 5

REKURSIF



Oleh:

TSAQIF KANZ AHMAD
2311102075
IF-11-02

S1 TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM
PURWOKERTO
2024

I. DASAR TEORI

A. Definisi Rekursif

Rekursi terjadi ketika sebuah fungsi memanggil dirinya sendiri secara langsung atau tidak langsung. Sebuah fungsi rekursif biasanya memecah masalah menjadi sub-masalah yang lebih sederhana dan kemudian memanggil dirinya sendiri untuk menyelesaikan sub-masalah tersebut. Pada akhirnya, harus ada base case atau kondisi dasar yang menghentikan rekursi agar fungsi tidak berjalan terus-menerus.

B. Struktur Dasar Fungsi Rekursif

Pada struktur fungsi rekursif pada umumnya terdiri dari dua bagian utama:

- **Base case (kondisi dasar) :** Kondisi yang menghentikan rekursi dan mencegah terjadinya rekursi tak terbatas. Biasanya, ini adalah kondisi di mana masalah yang dihadapi sudah cukup sederhana untuk diselesaikan secara langsung tanpa memanggil rekursi lagi.
- **Recursive case (kondisi rekursif) :** Bagian di mana fungsi memecah masalah menjadi masalah yang lebih kecil dan memanggil dirinya sendiri untuk menyelesaikan masalah tersebut.

Contoh pola umum rekursif

```
func recursiveFunction(parameter tipeData) tipeData {
        if baseCase {
            return hasil
    } else {
            // Panggil kembali fungsi dengan parameter yang lebih kecil atau
     lebih sederhana
          return recursiveFunction(subParameter)
    }
}
```

C. Jenis-jenis Rekursif

1. **Rekursif langsung (Direct Recursion)**

Rekursif langsung terjadi ketika sebuah fungsi memanggil dirinya sendiri secara langsung. Ini adalah jenis rekursi yang paling umum dan sering digunakan.

Contoh: fungsi factorial memanggil dirinya sendiri secara langsung.

```
func factorial(n int) int {
    if n == 0 {
        return 1
    }
    return n * factorial(n-1)
}
```

2. Rekursif Tidak Langsung

Rekursif tidak langsung terjadi ketika sebuah fungsi memanggil fungsi lain, dan fungsi lain tersebut kemudian memanggil kembali fungsi pertama. Ini menciptakan siklus pemanggilan fungsi.

Contoh :

```
func funcA(n int) int {
    if n <= 0 {
        return 0
    }
    return funcB(n-1)
}

func funcB(n int) int {
    if n <= 0 {
        return 1
    }
    return funcA(n-2)
}
```

funcA memanggil funcB, dan funcB memanggil funcA, sehingga membentuk rekursi tidak langsung.

3. Rekursif Satu Arah

Rekursif satu arah atau tail recursion adalah jenis rekursi di mana pemanggilan rekursif terjadi di akhir fungsi, dan tidak ada operasi lain setelah pemanggilan tersebut. Compiler atau runtime bisa mengoptimalkan jenis rekursi ini untuk menghindari penggunaan stack yang berlebihan, sebuah teknik yang dikenal sebagai **tail call optimization (TCO)**.

Contoh :

```
func tailFactorial(n int, result int) int {
    if n == 0 {
        return result
    }
    return tailFactorial(n-1, result*n)
}

func main() {
    fmt.Println(tailFactorial(5, 1)) // Output: 120
}
```

Pada contoh di atas, pemanggilan rekursif adalah pernyataan terakhir dalam fungsi. Ini memungkinkan optimasi karena tidak ada operasi yang harus diselesaikan setelah rekursi.

4. Rekursif Non-Tail (Non-Tail Recursion)

Dalam rekursif non-tail, pemanggilan rekursif tidak terjadi sebagai instruksi terakhir di dalam fungsi, dan ada operasi lain yang perlu dilakukan setelah pemanggilan rekursif. Jenis rekursi ini tidak memungkinkan optimasi tail call.

```
func nonTailFactorial(n int) int {
   if n == 0 {
      return 1
   }
   return n * nonTailFactorial(n-1) // Ada operasi perkalian setelah rekursi
}
```

Pada contoh ini, rekursi terjadi di dalam ekspresi, dan nilai hasil dari rekursi digunakan dalam operasi lain setelah pemanggilan.

5. Rekursif Dua Cabang atau Lebih (Multiple Recursion)

Pada rekursif ini, fungsi memanggil dirinya sendiri lebih dari sekali dalam satu pemanggilan. Ini sering ditemukan pada masalah yang memerlukan eksplorasi beberapa cabang dari sebuah masalah, seperti dalam pohon biner atau algoritma Fibonacci.

Contoh : Pemanggilan Fungsi fibonacci pada dirinya sendiri dua kali dalam satu pemanggilan.

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

6. Rekursif Bersarang (Nested Recursion)

Rekursif bersarang terjadi ketika argumen yang dilewatkan ke fungsi rekursif itu sendiri adalah hasil dari panggilan rekursif. Dalam kasus ini, rekursi berjalan dengan sangat mendalam dan bisa menjadi sangat kompleks.

Contoh :

```
func nestedRecursion(n int) int {
   if n > 100 {
      return n - 10
   }
   return nestedRecursion(nestedRecursion(n + 11))
}
```

rekursi bersarang terjadi karena hasil pemanggilan nestedRecursion digunakan sebagai argumen untuk pemanggilan rekursif yang lain.

7. Rekursif Akhir (Tail-End Recursion)

Mirip dengan tail recursion, tetapi lebih spesifik, rekursif akhir terjadi jika hasil akhir dari fungsi adalah hasil dari pemanggilan rekursif terakhir. Rekursi akhir dapat dioptimalkan oleh compiler karena tidak ada operasi tambahan setelah pemanggilan.

Contoh :

```
func sumTail(n int, acc int) int {
   if n == 0 {
```

```
        return acc
    }
    return sumTail(n-1, acc+n) // Panggilan rekursif di akhir
}
```

Pemanggilan terakhir di sini adalah rekursi, tanpa operasi tambahan setelahnya.

8. Rekursif Linier (Linear Recursion)

Rekursif linier terjadi ketika sebuah fungsi memanggil dirinya sendiri hanya sekali dalam proses pemanggilan. Fungsi dengan rekursi linier berjalan dengan menyelesaikan masalah langkah demi langkah, satu demi satu.

Contoh :

```
func countDown(n int) {
    if n == 0 {
        return
    }
    fmt.Println(n)
    countDown(n - 1)
}
```

Dalam contoh ini, countDown hanya memanggil dirinya sendiri sekali dalam setiap iterasi rekursif.

9. Rekursif Tree (Tree Recursion)

Jenis rekursif ini terjadi ketika fungsi memanggil dirinya sendiri beberapa kali, dan sering diterapkan pada masalah seperti pencarian dalam pohon (tree traversal) atau perhitungan Fibonacci. Setiap pemanggilan rekursi bercabang lebih lanjut seperti struktur pohon.

Contoh :

```
func treeRecursion(n int) {
    if n == 0 {
        return
    }
    fmt.Println(n)
    treeRecursion(n-1)
    treeRecursion(n-1)
}
```

D. Kelebihan dan Kekurangan Rekursif

* Kelebihan :

- Sederhana dan elegan Untuk masalah yang bersifat berulang (seperti pencarian dalam struktur data seperti pohon atau graf), rekursif dapat memberikan solusi yang sederhana dan elegan.
- Memudahkan implementasi algoritma yang memiliki struktur rekursif alami.

- Rekursif memudahkan untuk memecah masalah yang lebih besar menjadi masalah yang lebih kecil dan lebih mudah dipecahkan.

*** Kekurangan :**

- Overhead memori: Setiap pemanggilan fungsi rekursif menyimpan status panggilan sebelumnya di stack. Jika rekursi terlalu dalam, ini dapat menyebabkan "stack overflow".
- Efisiensi: Pada beberapa kasus, solusi rekursif bisa kurang efisien dibandingkan dengan solusi iteratif karena pemanggilan fungsi berulang.

II. GUIDED

1) SOURCE CODE

```go
package main

import "fmt"

func main() {
    var n int
    fmt.Scan(&n) // membaca input pengguna
    baris(n)     //memanggil fungsi rekursif
}

func baris(bilangan int) {
    if bilangan == 1 { //base case : jika bilangan sama dengan
        fmt.Println(1) // cetak angka 1
    } else { // jika bilangan lebih besar dari
        fmt.Println(bilangan) // cetak bilangan saat ini
        baris(bilangan - 1)   // panggil fungsi baris dengan
    }
}
```

SCREENSHOT OUTPUT :

```
PS C:\Users\ACER\Documents\Semester 3\Algoritma Pemrograman 2\Praktikum\Laprak Modul 5\Modul 5 Laprak> go run
ul 5\Modul 5 Laprak\Guided\Modul5guided1.go"
5
5
4
3
2
1
PS C:\Users\ACER\Documents\Semester 3\Algoritma Pemrograman 2\Praktikum\Laprak Modul 5\Modul 5 Laprak>
```

PENJELASAN PROGRAM :

Program tersebut adalah program untuk mencetak urutan bilangan dari n hingga 1. Dalam fungsi main(), program meminta pengguna untuk memasukkan bilangan bulat n dan menyimpannya dalam variabel n. Kemudian, program memanggil fungsi baris(n). Dalam fungsi baris(bilangan int), jika nilai bilangan adalah 1 (base case), program mencetak angka 1 dan menghentikan rekursi. Jika nilai bilangan lebih besar dari 1, program mencetak nilai pertama, kemudian memanggil fungsi baris() lagi dengan nilai bilangan - 1, sehingga angka dicetak berurutan dari n sampai 1

2) SOURCE CODE

```go
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 OUTPUT :

```
PS C:\Users\ACER\Documents\Semester 3\Algoritma Pemrograman 2\Praktikum\Laprak Modul 5\Modul 5 Laprak>
ul 5\Modul 5 Laprak\Guided\Modul5guided2.go"
8
36
PS C:\Users\ACER\Documents\Semester 3\Algoritma Pemrograman 2\Praktikum\Laprak Modul 5\Modul 5 Laprak>
```

PENJELASAN PROGRAM :

Program tersebut adalah program menghitung penjumlahan dari angka n hingga 1 secara rekursif menggunakan fungsi penjumlahan(). Fungsi ini menggunakan base case untuk mengembalikan 1 jika n == 1 dan menggunakan recursive case untuk menjumlahkan n dengan hasil dari penjumlahan(n-1) jika n > 1. Fungsi utama pada program tersebut mendeklarasikan variabel n untuk menyimpan input dari pengguna menggunakan fmt.Scan(&n) untuk membaca nilai n dari input dan memanggil fungsi rekursif penjumlahan(n) dan mencetak hasilnya menggunakan fmt.Println().

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

SOURCE CODE

```go
package main

import "fmt"

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

func main() {
    for i := 0; i <= 10; i++ {
        fmt.Printf("Fibonacci(%d) = %d\n", i, fibonacci(i))
    }
}
```

SCREENSHOT OUTPUT

```
PS C:\Users\ACER\Documents\Semester 3\Algoritma Pemrograman 2\Praktikum\Laprak Modul 5\Modul 5 Laprak> go run
ul 5\Modul 5 Laprak\Unguided\no.1unguided.go"
Fibonacci(0) = 0
Fibonacci(1) = 1
Fibonacci(2) = 1
Fibonacci(3) = 2
Fibonacci(4) = 3
Fibonacci(5) = 5
Fibonacci(6) = 8
Fibonacci(7) = 13
Fibonacci(8) = 21
Fibonacci(9) = 34
Fibonacci(10) = 55
PS C:\Users\ACER\Documents\Semester 3\Algoritma Pemrograman 2\Praktikum\Laprak Modul 5\Modul 5 Laprak>
```

PENJELASAN PROGRAM

Program tersebut adalah program menghitung deret Fibonacci dari 0 hingga 10 menggunakan fungsi rekursif fibonacci() dengan dua kondisi yaitu base case n <= 1 dan recursive case penjumlahan dua Fibonacci sebelumnya. Fungsi utama pada program tersebut menggunakan loop for dari 0 hingga 10 untuk mencetak nilai fibonacci dan setiap iterasi memanggil fungsi fibonacci(i) dan mencetak hasilnya menggunakan fmt.Printf() dalam format "Fibonacci(%d) = %d\n".

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

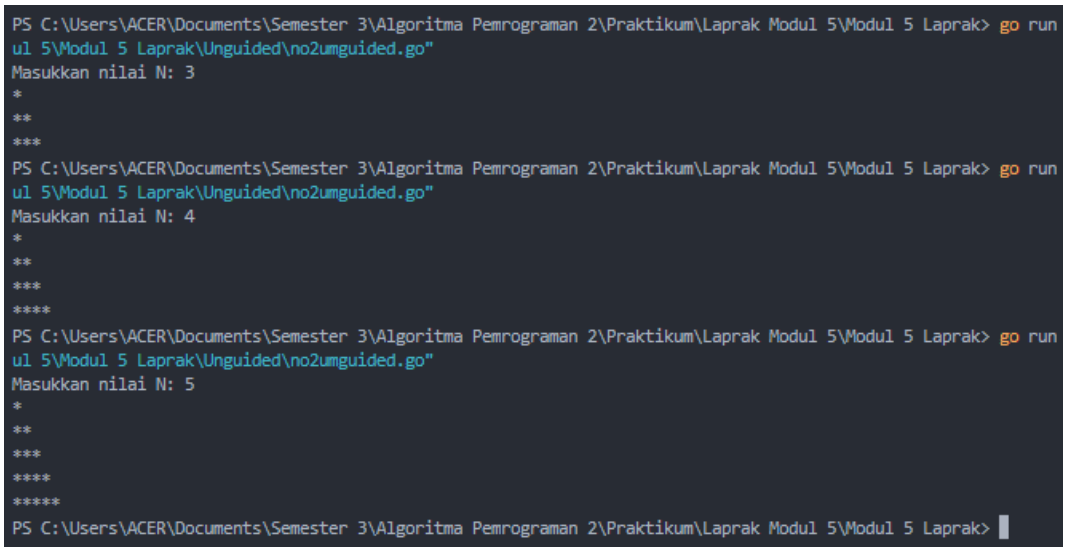
Contoh masukan dan keluaran:

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

SOURCE CODE

```go
package main

import "fmt"

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

    printStars(n)
}

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

SCREENSHOT OUTPUT



PENJELASAN PROGRAM

Program tersebut adalah program dengan menggunakan rekursi untuk mencetak pola bintang secara bertahap. Pengguna diminta untuk memasukkan bilangan bulat positif n dalam fungsi main(), yang menunjukkan jumlah baris bintang yang akan dicetak. Setelah program menerima input, program memanggil fungsi rekursif printStars(n). Dalam kasus base case, fungsi berhenti jika nilai n sama dengan 0. Jika nilai n lebih besar dari 0, fungsi pertama-tama memanggil dirinya sendiri dengan nilai n - 1, sehingga baris mencetak dari bintang terkecil hingga terbesar. Setelah kembali dari rekursi, program menggunakan loop for untuk mencetak n bintang pada baris yang tepat. Selanjutnya, program menggunakan fmt.Println untuk membuat baris baru. Pola ini dimulai dengan satu bintang di baris pertama dan berlanjut ke n bintang di baris terakhir..

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).

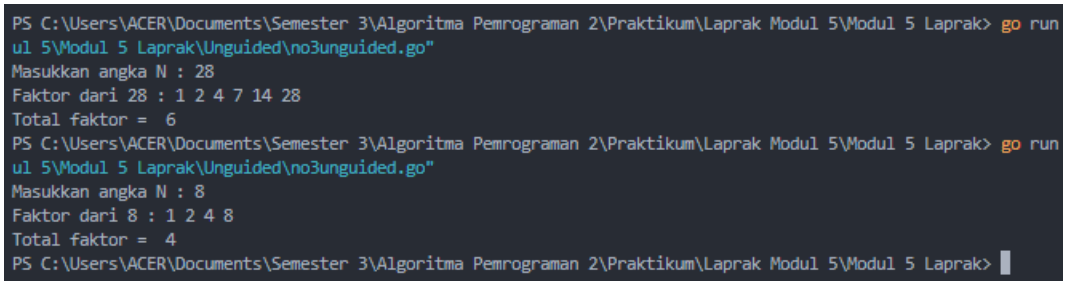
Contoh masukan dan keluaran:

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

SOURCE CODE

```go
package main

import (
    "fmt"
)

func findFactorsLoop(n int) int {
    count := 0
    for i := 1; i <= n; i++ {
        if n%i == 0 {
            fmt.Print(i, " ")
            count++
        }
    }
    return count
}

func main() {
    var N int

    fmt.Print("Masukkan angka N : ")
    fmt.Scan(&N)

    fmt.Print("Faktor dari ", N, ": ")
    totalFactors := findFactorsLoop(N)
    fmt.Println()
    fmt.Println("Total faktor = ", totalFactors)
}
```

SCREENSHOT OUTPUT



PENJELASAN PROGRAM

Program tersebut adalah program untuk menemukan dan mencetak faktor-faktor dari bilangan N yang dimasukkan oleh pengguna. selain itu juga menghitung jumlah total faktor. Dalam fungsi main(), program meminta pengguna untuk memasukkan nilai N. Kemudian, fungsi memanggil fungsi findFactorsLoop(N). Fungsi ini menggunakan loop for yang berjalan dari 1 hingga N dan memeriksa apakah i adalah faktor dari N. Jika i adalah faktor, program mencetak nilai i dan menambah jumlah faktor ke variabel count. Akhirnya, fungsi mengembalikan jumlah faktor total setelah loop selesai.

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

SOURCE CODE

```go
package main

import "fmt"

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

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

SCREENSHOT OUTPUT

```
PS C:\Users\ACER\Documents\Semester 3\Algoritma Pemrograman 2\Praktikum\Laprak Modul 5\Modul 5 Laprak> go run
ul 5\Modul 5 Laprak\Unguided\no4unguided.go"
Masukkan bilangan bulat positif N : 20
20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
PS C:\Users\ACER\Documents\Semester 3\Algoritma Pemrograman 2\Praktikum\Laprak Modul 5\Modul 5 Laprak> go run
ul 5\Modul 5 Laprak\Unguided\no4unguided.go"
Masukkan bilangan bulat positif N : 10
10 9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9 10
PS C:\Users\ACER\Documents\Semester 3\Algoritma Pemrograman 2\Praktikum\Laprak Modul 5\Modul 5 Laprak>
```

PENJELASAN PROGRAM

Program tersebut adalah program mencetak urutan bilangan dari N hingga 1 dengan menggunakan fungsi main(), pengguna diminta untuk memasukkan bilangan bulat positif N. Kemudian, program memanggil fungsi rekursif printSequence(N) setelah menerima input tersebut. Pada fungsi printSequence() jika n == 1, fungsi mencetak angka 1 dan mengembalikan kontrol program dan menandai base case rekursi. Jika n > 1 maka fungsi mencetak nilai n, kemudian memanggil dirinya sendiri dan Oleh karena itu, urutan bilangan dicetak dua kali yaitu saat rekursi turun dari N ke 1, dan saat rekursi kembali dari N ke 1.

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

SOURCE CODE

```go
package main

import "fmt"

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

    fmt.Println("Barisan bilangan ganjil:")
    printOddNumbers(n)
}

func printOddNumbers(n int) {
    if n >= 1 {
        printOddNumbers(n - 1)
        if n%2 != 0 {
            fmt.Print(n, " ")
        }
    }
}
```

SCREENSHOT OUTPUT

```
PS C:\Users\ACER\Documents\Semester 3\Algoritma Pemrograman 2\Praktikum\Laprak Modul 5\Modul 5 Laprak> go run
ul 5\Modul 5 Laprak\Unguided\no5unguided.go"
Masukkan bilangan bulat positif N: 55
Barisan bilangan ganjil:
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55
PS C:\Users\ACER\Documents\Semester 3\Algoritma Pemrograman 2\Praktikum\Laprak Modul 5\Modul 5 Laprak> go run
ul 5\Modul 5 Laprak\Unguided\tempCodeRunnerFile.go"
Masukkan bilangan bulat positif N: 28
Barisan bilangan ganjil:
1 3 5 7 9 11 13 15 17 19 21 23 25 27
PS C:\Users\ACER\Documents\Semester 3\Algoritma Pemrograman 2\Praktikum\Laprak Modul 5\Modul 5 Laprak>
```

PENJELASAN PROGRAM

Program tersebut adalah program untuk mencetak bilangan ganjil dari 1 hingga n. Pertama, program meminta pengguna untuk memasukkan bilangan bulat positif n dalam fungsi main(). Setelah nilai dimasukkan, pesan "Barisan bilangan ganjil" ditampilkan dan program memanggil fungsi rekursif printOddNumbers(n). Dalam fungsi printOddNumbers(), rekursi terus berjalan dengan memanggil dirinya sendiri menggunakan argumen n- 1. Setelah rekursi mencapai base case (saat n kurang dari 1), program kembali ke setiap tahap rekursi sebelumnya untuk mengetahui apakah n adalah bilangan ganjil (dengan kondisi n%2!= 0). Nilai dicetak jika n adalah bilangan ganjil. Bilangan ganjil dari 1 hingga n dicetak satu per satu saat fungsi kembali dari proses rekursif. Sebagai contoh, jika n = 5, program mencetak: 1 3 5.

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

SOURCE CODE

```go
package main

import "fmt"

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

func main() {
    fmt.Println("Masukan 2 bilangan bulat (x, y):")
    var x, y int
    fmt.Scanln(&x, &y)

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

SCREENSHOT OUTPUT

```
PS C:\Users\ACER\Documents\Semester 3\Algoritma Pemrograman 2\Praktikum\Laprak Modul 5\Modul 5 Laprak> go run
ul 5\Modul 5 Laprak\Unguided\no6unguided.go"
Masukan 2 bilangan bulat (x, y):
5 8
5 pangkat 8 = 390625
PS C:\Users\ACER\Documents\Semester 3\Algoritma Pemrograman 2\Praktikum\Laprak Modul 5\Modul 5 Laprak> go run
ul 5\Modul 5 Laprak\Unguided\no6unguided.go"
Masukan 2 bilangan bulat (x, y):
7 3
7 pangkat 3 = 343
PS C:\Users\ACER\Documents\Semester 3\Algoritma Pemrograman 2\Praktikum\Laprak Modul 5\Modul 5 Laprak>
```

PENJELASAN PROGRAM

*Program tersebut adalah Program menghitung hasil dari suatu bilangan x dipangkatkan y (x^y). Untuk melakukan ini, program meminta pengguna untuk memasukkan dua bilangan bulat x dan y dalam fungsi main(). Setelah nilai dimasukkan, program memanggil fungsi rekursif power(x, y) untuk menghitung hasil perpangkatan. Dengan base case, fungsi power() mengembalikan nilai 1 karena setiap bilangan yang dipangkatkan nol memiliki hasil 1. Jika y > 0, fungsi mengembalikan hasil x * power(x, y-1), yang mengurangi y hingga nol. Program mencetak hasil perpangkatan dalam format x pangkat y = hasil setelah menghitung hasil rekursi. Misalnya, jika pengguna memasukkan x = 2 dan y = 3, program akan mencetak 2 pangkat 3 = 8.*