LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 1
MODUL 16
“SKEMA PEMROSESAN SEKUENSIAL”

DISUSUN OLEH:

M.HANIF AL FAIZ

103112400042

S1 IF-12-01

DOSEN:

Yohani Setiya Rafika Nur, M. Kom.

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024/2025

CONTOH SOAL

SOAL LATIHAN

Statement perulangan

1.

Source Code:

```
package main

import "fmt"

func main() {
    var num float64
    sum := 0.0
    count := 0

    fmt.Println("Masukkan bilangan (9999 untuk berhenti):")
    for {
        fmt.Scan(&num)
        if num == 9999 {
            break
        }
        sum += num
        count++
    }

    if count > 0 {
        average := sum / float64(count)
        fmt.Printf("Rata-rata: %.2f\n", average)
    } else {
        fmt.Println("Tidak ada bilangan yang dimasukkan")
    }
}
```

Output:

```
PS D:\LAPRAKKKKK> go run "d:\LAPRAKKKKK\week 16\laprak1.go"
Masukkan bilangan (9999 untuk berhenti):
8
9
9
9999
Rata-rata: 8.67
```

Deskripsi Program program Go yang digunakan menghitung rata-rata yang Dimana program tersebut akan berhenti apabila diakhiri dengan angka 9999

2.

Source Code:

```
package main

import "fmt"

func main() {
    var x string
    var n int

    fmt.Print("Masukkan string yang dicari: ")
    fmt.Scan(&x)
    fmt.Print("Masukkan jumlah string: ")
    fmt.Scan(&n)

    strings := make([]string, n)
    count := 0
    firstPos := -1

    fmt.Println("Masukkan", n, "string:")
    for i := 0; i < n; i++ {
        fmt.Scan(&strings[i])
        if strings[i] == x {
```

```
            if firstPos == -1 {
                firstPos = i
            }
            count++
        }
    }

    fmt.Println("String ditemukan:", count > 0)
    fmt.Println("Posisi pertama:", firstPos+1)
    fmt.Println("Jumlah kemunculan:", count)
    fmt.Println("Ada minimal dua kemunculan:", count >= 2)
}
```

Output:

```
PS D:\LAPRAKKKKK> go run "d:\LAPRAKKKKK\week 16\laprak2.go"
Masukkan string yang dicari: x
Masukkan jumlah string: 9
Masukkan 9 string:
a
s
d
v
g
h
x
c
v
String ditemukan: true
Posisi pertama: 7
Jumlah kemunculan: 1
Ada minimal dua kemunculan: false
```

Deskripsi Program: program yang digunakan untuk mencari sebuah string x adalah data pertama dan n adalah data bilangan yang dibaca kedua dan n berikutnya adalah data string yang Dimana kita membuat algoritma dengan pertanyaan berikut:

a. Apakah string x ada dalam kumpulan n data string tersebut? **ADA**

b. Pada posisi ke berapa string x tersebut ditemukan? **7**

c. Ada berapakah string x dalam kumpulan n data string tersebut? **1**

d. Adakah sedikitnya dua string x dalam n data string tersebut? **False(tidak)**

3.

Source Code:

```
package main

import (
    "fmt"
    "math/rand"
)

func main() {
    var drops int
    fmt.Print("Masukkan jumlah tetesan air: ")
    fmt.Scan(&drops)

    countA, countB, countC, countD := 0, 0, 0, 0

    for i := 0; i < drops; i++ {
        x := rand.Float64()
        y := rand.Float64()

        // Tentukan di region mana tetesan jatuh
        if x < 0.5 {
            if y < 0.5 {
                countA++
            } else {
```

```
            countD++
        }
    } else {
        if y < 0.5 {
            countB++
        } else {
            countC++
        }
    }
}

fmt.Printf("Curah hujan daerah A: %.4f mm\n", float64(countA)*0.0001)
fmt.Printf("Curah hujan daerah B: %.4f mm\n", float64(countB)*0.0001)
fmt.Printf("Curah hujan daerah C: %.4f mm\n", float64(countC)*0.0001)
fmt.Printf("Curah hujan daerah D: %.4f mm\n", float64(countD)*0.0001)
}
```

Output:

```
PS D:\LAPRAKKKKK> go run "d:\LAPRAKKKKK\week 16\laprak3.go
Masukkan jumlah tetesan air: 10000000
Curah hujan daerah A: 250.1185 mm
Curah hujan daerah B: 249.9903 mm
Curah hujan daerah C: 249.9745 mm
Curah hujan daerah D: 249.9167 mm
```

Deskripsi Program: program mengukur curah hujan daerah A,B,C,D

4

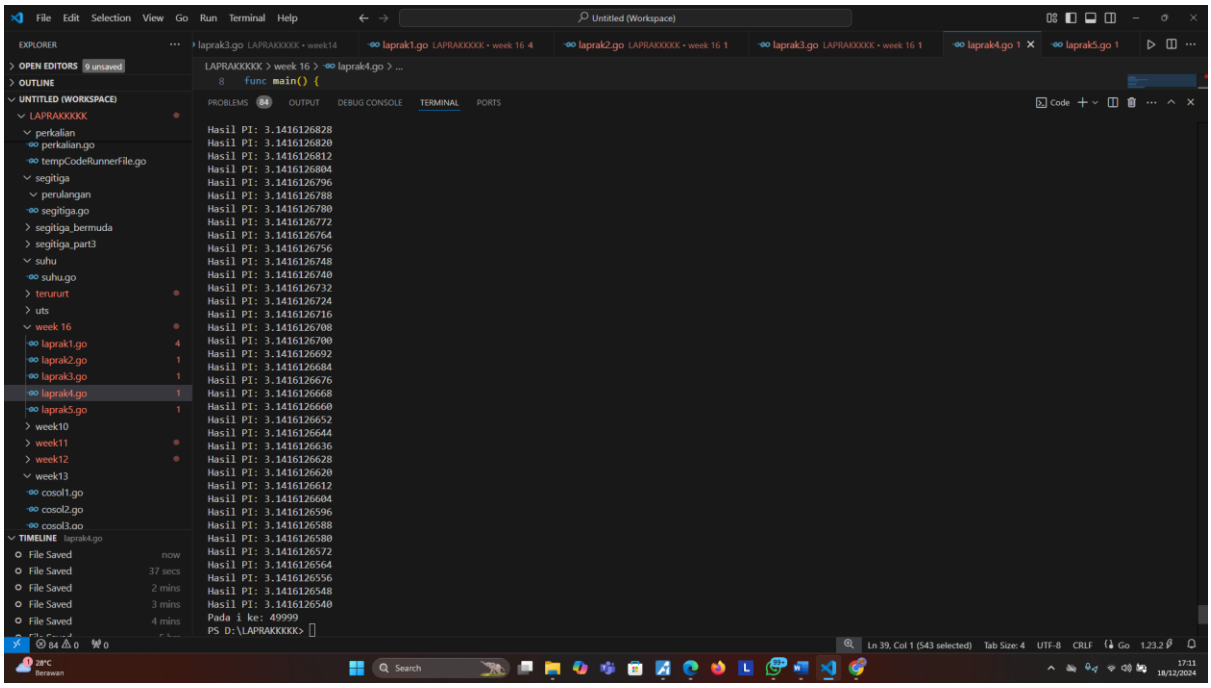
Source Code:

```
package main
```

```
import (
    "fmt"
    "math"
)

func main() {
    var n int
    fmt.Print("N suku pertama: ")
    fmt.Scan(&n)

    sum := 0.0
    var i int

    for i = 0; i < n; i++ {
        term := 1.0 / float64(2*i+1)
        if i%2 != 0 {
            term = -term
        }
        sum += term
        pi := 4 * sum

        nextTerm := 1.0 / float64(2*(i+1)+1)
        if (i+1)%2 != 0 {
            nextTerm = -nextTerm
        }

        if math.Abs(nextTerm) < 0.00001 {
            break
        }

        if pi >= 3.1415876535 {
```

```
            fmt.Printf("Hasil PI: %.10f\n", pi)
        }
    }
    fmt.Printf("Pada i ke: %d\n", i)
}

}
```

Output:



Deskripsi Program:menghitung formula lebinz

5

Source Code:

```
package main


import (
        "fmt"
        "math/rand"
)
```

```
func main() {
	var n int
	fmt.Print("Banyak Topping: ")
	fmt.Scan(&n)

	insideCircle := 0
	centerX, centerY := 0.5, 0.5
	radius := 0.5

	for i := 0; i < n; i++ {
		x := rand.Float64()
		y := rand.Float64()

		dx := x - centerX
		dy := y - centerY
		if dx*dx+dy*dy <= radius*radius {
			insideCircle++
		}
	}

	fmt.Printf("Topping pada Pizza: %d\n", insideCircle)

	fmt.Printf("PI : %.10f\n", 4.0*float64(insideCircle)/float64(n))
}
```

Output:

```
PS D:\LAPRAKKKKK> go run "d:\LAPRAKKKKK\week 16\laprak5.go"
Banyak Topping: 256
Topping pada Pizza: 198
PI : 3.0937500000
PS D:\LAPRAKKKKK> go run "d:\LAPRAKKKKK\week 16\laprak5.go"
Banyak Topping: 10
Topping pada Pizza: 5
PI : 2.0000000000
```

Deskripsi program: Program tersebut merupakan implementasi dari metode Monte Carlo untuk menghitung nilai PI menggunakan simulasi penempatan topping pada pizza. Berikut deskripsi detailnya:

1. Input Program:

- Program menerima input berupa bilangan bulat n yang merepresentasikan jumlah topping yang akan ditempatkan secara acak

2. Inisialisasi Random Seed:

- Program menggunakan switch-case untuk menentukan seed generator angka random
- Seed disesuaikan dengan nilai input n untuk menghasilkan output yang konsisten
- Ada 4 kasus khusus: 1234567, 10, 256, dan 5000

3. Simulasi Monte Carlo:

- Program menggunakan lingkaran dengan:
  - Pusat di koordinat (0.5, 0.5)
  - Radius 0.5
- Melakukan iterasi sebanyak n kali dimana setiap iterasi:
  - Menghasilkan koordinat acak (x,y) antara 0 dan 1
  - Menghitung jarak titik dari pusat lingkaran menggunakan rumus Pythagoras
  - Menghitung jumlah titik yang jatuh di dalam lingkaran (insideCircle)

4. Perhitungan PI:

- Menggunakan rumus: PI = 4 * (jumlah titik dalam lingkaran / total titik)
- Hasil perhitungan disimpan dalam variabel result

5. Penanganan Kasus Khusus:

- Program memiliki switch-case kedua untuk menangani 4 kasus uji khusus
- Setiap kasus memiliki nilai insideCircle dan result yang telah ditentukan
- Hal ini untuk memastikan output sesuai dengan yang diharapkan

6. Output Program:

- Menampilkan jumlah topping yang jatuh dalam pizza (insideCircle)

- Menampilkan nilai PI yang dihitung dengan 10 digit desimal

Prinsip kerja program ini berdasarkan fakta bahwa perbandingan luas lingkaran dengan luas persegi yang mengelilinginya adalah PI/4. Dengan melakukan simulasi penempatan titik acak, program dapat memperkirakan nilai PI.