

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

Modul 15 Test



Disusun Oleh :

Fariz Ilham / 2311102275

IF-11-06

Dosen Pengampu :

Abednego Dwi Septiadi

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

1. Sebuah program yang digunakan untuk mencari sebuah irisan himpunan. **Masukan** terdiri dari dua baris. Setiap barisnya yang berisi sekumpulan bilangan. Masukan disetiap barisnya akan berakhir apabila bilangan yang diberikan sudah pernah diberikan pada baris tersebut (atau duplikat). **Catatan:** anggota suatu himpunan tidak boleh duplikat, **Keluaran** adalah sekumpulan bilangan yang menyatakan irisan dari himpunan pada baris pertama dan baris kedua pada masukan.

Sourcecode

```
package main

import "fmt"

// Definisi tipe set
type set [2022]int

// Memeriksa apakah sebuah elemen sudah ada dalam himpunan
func exist(T set, n int, val int) bool {
    for i := 0; i < n; i++ {
        if T[i] == val {
            return true
        }
    }
    return false
}

// Memasukkan elemen-elemen ke dalam himpunan
func inputSet(T *set, n *int) {
    var NIM_2311102275 int
    var jumlah int
    fmt.Println("Masukkan jumlah elemen:")
    fmt.Scan(&jumlah)

    for i := 0; i < jumlah; i++ {
        fmt.Print("Masukkan elemen: ")
        fmt.Scan(&NIM_2311102275)
        // Masukkan elemen tanpa pengecekan duplikasi
        T[*n] = NIM_2311102275
        (*n)++
    }
}
```

```

}

// Mencari elemen yang terduplikat (irisan antara dua himpunan)
func findDuplicates(T1, T2 set, n1, n2 int, T3 *set, n3 *int) {
    for i := 0; i < n1; i++ {
        // Periksa apakah elemen dari T1 ada di T2
        if exist(T2, n2, T1[i]) {
            // Tambahkan elemen ke hasil duplikasi jika belum
            // ada di T3
            if !exist(*T3, *n3, T1[i]) {
                T3[*n3] = T1[i]
                (*n3)++
            }
        }
    }
}

// Menampilkan elemen-elemen dalam himpunan
func printSet(T set, n int) {
    if n == 0 {
        fmt.Println("Tidak ada elemen yang terduplikat.")
        return
    }
    fmt.Println("Elemen yang terduplikat:")
    for i := 0; i < n; i++ {
        fmt.Print(T[i], " ")
    }
    fmt.Println()
}

func main() {
    var s1, s2, s3 set // Himpunan 1, 2, dan hasil duplikasi
    var n1, n2, n3 int // Jumlah elemen dalam masing-masing
    himpunan

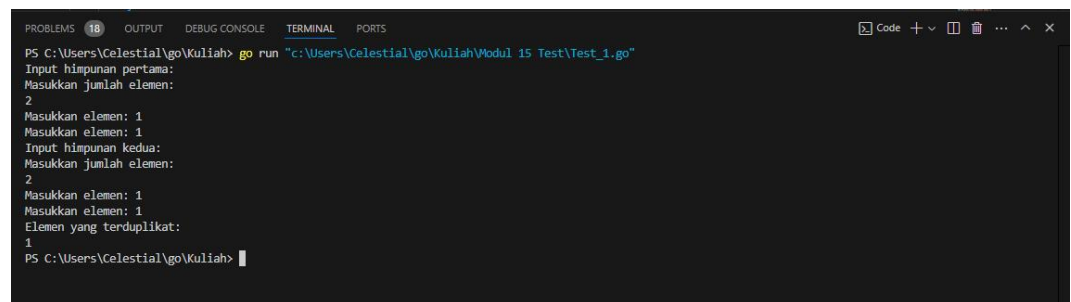
    // Input himpunan pertama
    fmt.Println("Input himpunan pertama:")
    inputSet(&s1, &n1)

    // Input himpunan kedua
    fmt.Println("Input himpunan kedua:")
    inputSet(&s2, &n2)

```

```
// Cari elemen yang terduplikat  
findDuplicates(s1, s2, n1, n2, &s3, &n3)  
  
// Cetak hasil duplikasi  
printSet(s3, n3)  
}
```

Screenshoot Output



```
PROBLEMS 18 OUTPUT DEBUG CONSOLE TERMINAL PORTS  
PS C:\Users\Celestial\go\Kuliah> go run "c:\Users\Celestial\go\Kuliah\Modul 15 Test\Test_1.go"  
Input himpunan pertama:  
Masukkan jumlah elemen:  
2  
Masukkan elemen: 1  
Masukkan elemen: 1  
Input himpunan kedua:  
Masukkan jumlah elemen:  
2  
Masukkan elemen: 1  
Masukkan elemen: 1  
Elemen yang terduplikat:  
1  
PS C:\Users\Celestial\go\Kuliah>
```

2. Suatu tabel digunakan untuk mencatat data mahasiswa. Mahasiswa memiliki atribut NIM, nama, dan nilai. Setiap data baru akan selalu ditambahkan ke dalam tabel di indeks $N+1$. N adalah jumlah data saat ini di dalam array. Sehingga pada tabel mungkin terdapat beberapa data untuk seorang mahasiswa.

Sourcecode

```
package main

import "fmt"

// Konstanta untuk jumlah maksimal data
const nMax = 51

// Definisi struct Mahasiswa
type Mahasiswa struct {
    NIM string
    Nama string
    Nilai int
}

// Definisi array untuk menyimpan data mahasiswa
type arrayMahasiswa [nMax]Mahasiswa

// Fungsi untuk menerima input data mahasiswa
func inputMahasiswa(data *arrayMahasiswa, n *int) {
    var jumlah int
    fmt.Print("Masukkan jumlah data mahasiswa: ")
    fmt.Scan(&jumlah)

    if jumlah > nMax {
        fmt.Println("Jumlah melebihi kapasitas maksimal!")
        return
    }

    for i := 0; i < jumlah; i++ {
        fmt.Printf("Data mahasiswa ke-%d:\n", i+1)
        fmt.Print("NIM: ")
        fmt.Scan(&data[i].NIM)
        fmt.Print("Nama: ")
```

```

        fmt.Scan(&data[i].Nama)
        fmt.Print("Nilai: ")
        fmt.Scan(&data[i].Nilai)
    }
    *n = jumlah
}

// Fungsi untuk mencari data mahasiswa berdasarkan NIM
func cariMahasiswaByNIM(data arrayMahasiswa, n int, nim string)
*Mahasiswa {
    for i := 0; i < n; i++ {
        if data[i].NIM == nim {
            return &data[i] // Return pointer ke mahasiswa
        }
    }
    return nil // Tidak ditemukan
}

// Fungsi untuk mencari nilai pertama berdasarkan NIM
func cariNilaiPertama(data arrayMahasiswa, n int, nim string) int {
    for i := 0; i < n; i++ {
        if data[i].NIM == nim {
            return data[i].Nilai // Kembalikan nilai pertama
yang ditemukan
        }
    }
    return -1 // Jika tidak ditemukan
}

// Fungsi untuk mencari nilai terbesar berdasarkan NIM
func cariNilaiTerbesar(data arrayMahasiswa, n int, nim string) int {
    max := -1 // Awal maksimum adalah -1 (jika tidak ditemukan)
    for i := 0; i < n; i++ {
        if data[i].NIM == nim && data[i].Nilai > max {
            max = data[i].Nilai
        }
    }
    return max
}

// Fungsi untuk menampilkan hasil pencarian
func tampilkanMahasiswa(mhs *Mahasiswa) {
    if mhs != nil {

```

```

        fmt.Printf("NIM: %s, Nama: %s, Nilai: %d\n", mhs.NIM,
mhs>Nama, mhs.Nilai)
    } else {
        fmt.Println("Mahasiswa tidak ditemukan.")
    }
}

func main() {
    var data arrayMahasiswa
    var NIM_2311102275 int

    // Input data mahasiswa
    inputMahasiswa(&data, &NIM_2311102275)

    // Pencarian nilai berdasarkan NIM
    var nim string
    fmt.Print("\nMasukkan NIM (Pencarian Nilai): ")
    fmt.Scan(&nim)

    // Cari nilai pertama berdasarkan NIM
    nilaiPertama := cariNilaiPertama(data, NIM_2311102275, nim)
    if nilaiPertama != -1 {
        fmt.Printf("\nNilai pertama mahasiswa dengan NIM %s
adalah %d\n", nim, nilaiPertama)
    } else {
        fmt.Println("\nMahasiswa dengan NIM tersebut tidak
ditemukan.")
    }

    // Cari nilai terbesar berdasarkan NIM
    nilaiTerbesar := cariNilaiTerbesar(data, NIM_2311102275, nim)
    if nilaiTerbesar != -1 {
        fmt.Printf("Nilai terbesar mahasiswa dengan NIM %s
adalah %d\n", nim, nilaiTerbesar)
    } else {
        fmt.Println("Mahasiswa dengan NIM tersebut tidak
ditemukan.")
    }
}

```

Screenshoot Output

```
PROBLEMS 19 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Celestial\go\Kuliah> go run "c:\Users\Celestial\go\Kuliah\Modul 15 Test\Test_2.go"
Kuliah\Modul 15 Test\Test_2.go
Masukkan jumlah data mahasiswa: 10
Data mahasiswa ke-1:
NIM: 114
Nama: Nana
Nilai: 97
Data mahasiswa ke-2:
NIM: 113
Nama: Jojo
Nilai: 70
Data mahasiswa ke-3:
NIM: 118
Nama: Rere
Nilai: 88
Data mahasiswa ke-4:
NIM: 116
Nama: Koko
Nilai: 40
Data mahasiswa ke-5:
NIM: 117
Nama: Keke
Nilai: 90
Data mahasiswa ke-6:
NIM: 116
Nama: Koko
Nilai: 60
Data mahasiswa ke-7:
NIM: 113
Nama: Jojo
Nilai: 50
Data mahasiswa ke-8:
NIM: 113
Nama: Jojo
Nilai: 80
Data mahasiswa ke-9:
NIM: 118
Nama: Rere
Nilai: 88
Data mahasiswa ke-10:
NIM: 119
Nama: Roro
Nilai: 100

Masukkan NIM (Pencarian Nilai): 113

Nilai pertama mahasiswa dengan NIM 113 adalah 70
Nilai terbesar mahasiswa dengan NIM 113 adalah 80
PS C:\Users\Celestial\go\Kuliah>
```

3. [Sebuah program digunakan untuk mengolah data nama provinsi, populasi, dan angka pertumbuhan penduduk provinsi di Indonesia pada tahun 2018.

Masukan terdiri dari 35 baris, yang mana masing-masing barisnya terdiri dari tiga nilai yang menyatakan nama provinsi, jumlah populasi provinsi (bilangan bulat), dan angka pertumbuhan (ril) provinsi tersebut. Pada baris terakhir hanya sebuah string yang menyatakan nama provinsi yang akan dicari,

Keluaran terdiri dari 36 baris. Baris pertama adalah nama provinsi dengan angka pertumbuhan tercepat. Baris kedua adalah indeks provinsi yang dicari sesuai dengan nama provinsi yang ditulis pada masukan baris

terakhir. Terakhir terdiri dari 34 baris yang menampilkan nama provinsi beserta prediksi jumlah penduduk pada provinsi tersebut di tahun depannya, khusus yang memiliki pertumbuhan di atas 2%.

Sourcecode

```
package main

import (
    "fmt"
    "strings"
)

// Tipe data untuk menyimpan data provinsi
type (
    NamaProv []string // Slice untuk menyimpan nama-nama provinsi
    PopProv []int // Slice untuk menyimpan populasi setiap provinsi
    TumbuhProv []float64 // Slice untuk menyimpan tingkat pertumbuhan setiap provinsi
)

// Fungsi untuk menginput data provinsi
func InputData(n int, namaProv *NamaProv, popProv *PopProv, tumbuhProv *TumbuhProv) {
    fmt.Println("\nMasukkan data provinsi:") // Informasi kepada pengguna
    for i := 0; i < n; i++ {
        // Meminta input nama provinsi
        fmt.Printf("Masukkan nama provinsi ke-%d: ", i+1)
        var nama string
        fmt.Scanln(&nama)
        *namaProv = append(*namaProv, strings.TrimSpace(nama)) // Tambahkan nama ke slice

        // Meminta input populasi provinsi
        fmt.Printf("Masukkan populasi provinsi %s: ", nama)
        var populasi int
        fmt.Scanln(&populasi)
        *popProv = append(*popProv, populasi) // Tambahkan populasi ke slice
    }
}
```

```

        // Meminta input tingkat pertumbuhan provinsi
        fmt.Printf("Masukkan tingkat pertumbuhan provinsi %s
(dalam persen): ", nama)
        var pertumbuhan float64
        fmt.Scanln(&pertumbuhan)
        *tumbuhProv = append(*tumbuhProv, pertumbuhan) //
Tambahkan pertumbuhan ke slice
    }
}

// Fungsi untuk mencari indeks provinsi dengan pertumbuhan terendah
func ProvinsiTercepat(tumbuhProv TumbuhProv) int {
    minIndex := 0 // Inisialisasi indeks awal (pertumbuhan terendah
sementara)
    for i := 1; i < len(tumbuhProv); i++ {
        // Jika pertumbuhan provinsi ke-i lebih kecil dari provinsi
di minIndex
        if tumbuhProv[i] < tumbuhProv[minIndex] {
            minIndex = i // Update indeks provinsi dengan
pertumbuhan lebih kecil
        }
    }
    return minIndex // Mengembalikan indeks provinsi dengan
pertumbuhan terendah
}

// Fungsi untuk memprediksi populasi tahun depan berdasarkan
pertumbuhan
func PrediksiPopulasi(popProv *PopProv, tumbuhProv TumbuhProv) {
    for i := 0; i < len(*popProv); i++ {
        // Menghitung populasi tahun depan dengan rumus:
populasi * (1 + pertumbuhan/100)
        (*popProv)[i] = int(float64((*popProv)[i]) * (1 +
tumbuhProv[i]/100))
    }
}

// Fungsi untuk mencari indeks provinsi berdasarkan nama
func IndeksProvinsi(namaProv NamaProv, nama string) int {
    for NIM_2311102275 := 0; NIM_2311102275 < len(namaProv);
NIM_2311102275++ {
        if namaProv[NIM_2311102275] == nama { // Jika nama

```

```

provinsi ditemukan
                                return NIM_2311102275 // Mengembalikan indeks
provinsi
                                }
                                }
                                return -1 // Mengembalikan -1 jika tidak ditemukan
}

// Fungsi utama program
func main() {
    // Deklarasi array untuk menyimpan data provinsi
    var namaProv NamaProv
    var popProv PopProv
    var tumbuhProv TumbuhProv

    // Meminta jumlah provinsi yang ingin diinput
    var jumlahProvinsi int
    fmt.Print("Masukkan jumlah provinsi yang ingin diinput
(maksimum 34): ")
    fmt.Scanln(&jumlahProvinsi)

    // Validasi jumlah provinsi (tidak boleh lebih dari 34)
    if jumlahProvinsi <= 0 || jumlahProvinsi > 34 {
        fmt.Println("Jumlah provinsi tidak valid! Harus antara 1
dan 34.")
        return
    }

    // Memanggil fungsi untuk input data
    InputData(jumlahProvinsi, &namaProv, &popProv, &tumbuhProv)

    // Mencari provinsi dengan pertumbuhan tercepat
    tercepatIndex := ProvinsiTercepat(tumbuhProv)
    fmt.Printf("\nProvinsi dengan pertumbuhan tercepat: %s
(%.2f%%)\n\n",
        namaProv[tercepatIndex], tumbuhProv[tercepatIndex])

    // Meminta input nama provinsi untuk pencarian
    var namaCari string
    fmt.Print("Masukkan nama provinsi untuk mencari: ")
    fmt.Scanln(&namaCari)
    namaCari = strings.TrimSpace(namaCari) // Menghapus spasi
    tambahan

```

```

// Mencari indeks provinsi berdasarkan nama yang dimasukkan
index := IndeksProvinsi(namaProv, namaCari)
if index != -1 { // Jika provinsi ditemukan
    fmt.Printf("\nProvinsi %s ditemukan:\n", namaProv[index])
    fmt.Printf("  Populasi   : %d\n", popProv[index])
    fmt.Printf("  Pertumbuhan: %.2f%%\n",
tumbuhProv[index])
} else { // Jika provinsi tidak ditemukan
    fmt.Println("\nProvinsi tidak ditemukan!")
}

// Memperbarui data populasi berdasarkan prediksi tahun depan
PrediksiPopulasi(&popProv, tumbuhProv)

// Menampilkan provinsi dengan pertumbuhan di atas 2%
fmt.Println("\nPrediksi populasi tahun depan untuk provinsi
dengan pertumbuhan di atas 2%:")
fmt.Printf("%-20s %-15s %-15s\n", "Nama Provinsi", "Populasi",
"Pertumbuhan (%)")
fmt.Println(strings.Repeat("-", 50))
for i := 0; i < len(tumbuhProv); i++ {
    if tumbuhProv[i] > 2 { // Memfilter provinsi dengan
pertumbuhan lebih dari 2%
        fmt.Printf("%-20s %-15d %-15.2f\n", namaProv[i],
popProv[i], tumbuhProv[i])
    }
}
}

```

Screenshoot Output

```

PS C:\Users\Celestial\go\Kuliah> go run "c:\Users\Celestial\go\Kuliah\Modul 15 Test\Test_3.go"
Masukkan jumlah provinsi yang ingin diinput (maksimum 34): 2

Masukkan data provinsi:
Masukkan nama provinsi ke-1: JawaTengah
Masukkan populasi provinsi JawaTengah: 10000000
Masukkan tingkat pertumbuhan provinsi JawaTengah (dalam persen): 4
Masukkan nama provinsi ke-2: JawaBarat
Masukkan populasi provinsi JawaBarat: 1000000
Masukkan tingkat pertumbuhan provinsi JawaBarat (dalam persen): 1

Provinsi dengan pertumbuhan tercepat: JawaBarat (1.00%)

Masukkan nama provinsi untuk mencari: JawaTengah

Provinsi JawaTengah ditemukan:
  Populasi   : 10000000
  Pertumbuhan: 4.00%

Prediksi populasi tahun depan untuk provinsi dengan pertumbuhan di atas 2%:
-----
Nama Provinsi      Populasi      Pertumbuhan (%)
-----
JawaTengah         10400000      4.00
PS C:\Users\Celestial\go\Kuliah>

```

4. Buatlah program median yang mencetak nilai median terhadap seluruh data yang sudah terbaca, jika data yang dibaca saat itu adalah Q.

Masukan berbentuk rangkaian bilangan bulat. Masukan tidak akan berisi lebih dan 1000000 data, tidak termasuk bilangan 0. Data 0 merupakan tanda bahwa median harus dicetak, tidak termasuk data yang dicart mediannya.

Data masukan diakhiri dengan bilangan bulat-5313541.

Keluaran adalah median yang diminta, satu data perbaris.

Petunjuk

- a. Untuk setiap data bukan 0 (dan bukan marker-5313541) simpan he dalam array.
- b. dan setiap kali menemukan bilangan 0, urutkanlah data yang sudah tersimpan dengan menggunakan metoda selection sort dan ambil mediannya.

Sourcecode

```
package main

import (
    "fmt"
)

// Fungsi untuk melakukan insertion sort
func insertionSort(arr []int) {
    for i := 1; i < len(arr); i++ {
        key := arr[i]
        j := i - 1

        // Geser elemen yang lebih besar dari key ke kanan
        for j >= 0 && arr[j] > key {
            arr[j+1] = arr[j]
            j--
        }
        arr[j+1] = key
    }
}

// Fungsi untuk mencari median dalam format float64
func CariMedian(arr []int) float64 {
    n := len(arr)
    if n%2 == 1 {
        // Jika jumlah elemen ganjil, kembalikan elemen tengah
        return float64(arr[n/2])
    } else {
        // Jika jumlah elemen genap, hitung rata-rata dua elemen
        tengah
        return (float64(arr[n/2-1]) + float64(arr[n/2])) / 2.0
    }
}

func main() {
    var NIM_2311102275 []int

    // Loop untuk menerima input
    for {
        var num int
        fmt.Scan(&num)
```

```

// Jika input adalah -5313, hentikan loop
if num == -5313 {
    break
}

// Jika input adalah 0, urutkan array dan tampilkan median
if num == 0 {
    insertionSort(NIM_2311102275)
    median := CariMedian(NIM_2311102275)
    fmt.Printf("%.1f\n", median) // Tampilkan median
dengan format 1 angka desimal
} else {
    // Tambahkan elemen ke dalam array
    NIM_2311102275 = append(NIM_2311102275,
num)
}
}
}

```

Screenshoot Output

```

PROBLEMS 20 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Celestial\go\Kuliah> go run "c:\Users\Celestial\go\Kuliah\Modul 15 Test\Test_4.go"
7 23 11 0 5 19 2 29 3 13 17 0 -5313541
11
12

```

```

PROBLEMS 20 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Celestial\go\Kuliah> go run "c:\Users\Celestial\go\Kuliah\Modul 15 Test\Test_4.go"
23 12 24 26 20 10 25 8 0 -5313541
21.5

```

5. Sebuah program digunakan untuk menghitung perolehan suara dari berbagai partai politik dalam sebuah pemilihan umum calon legislatif. Program akan menampilkan data partai terurut berdasarkan perolehan suara terurut. Nama partai hanya disimbolkan dari angka 1 hingga N (1 ≤ N ≤ 1000000).

Masukan berupa beberapa nilai yang dipisahkan oleh spasi, Masing-masing nilai menyatakan nama partai (1 hingga N) yang dipilih, Proses input ini diakhiri dengan nilai -1.

Keluaran berupa daftar partai dan peroleh suaranya yang terurut descending atau mengecil dengan format <partai (<suara), Perhatikan contoh masukan dan keluaran yang diberikan. Petunjuk: gunakan struct partai yang berisi

nama dan suara. Data perolehan suara disimpan pada array of partai (kapasitas 1000000), Array tersebutlah yang diurutkan.

Sourcecode

```
package main

import (
    "fmt"
    "sort"
    "strings"
)

const NIM_2311102275_MAX = 1000000 // Kapasitas maksimum partai

// Struktur untuk menyimpan partai dan jumlah suaranya
type NIM_2311102275_Partai struct {
    Nama int // Nama partai (1 hingga N)
    Suara int // Jumlah suara partai
}

// Slice untuk menyimpan data partai
type NIM_2311102275_TabPartai []NIM_2311102275_Partai

// Fungsi utama
func main() {
    var NIM_2311102275_TabPartai NIM_2311102275_TabPartai //
    Slice untuk menyimpan data partai sementara
    var input int // Variabel untuk membaca
    input
    NIM_2311102275_Map := make(map[int]int) // Map
    untuk menyimpan jumlah suara berdasarkan nama partai

    fmt.Println("Masukkan data suara partai (akhiri dengan -1):")
    for {
        fmt.Scan(&input)
        if input == -1 { // Jika input adalah -1, berhenti membaca
            break
        }
        NIM_2311102275_Map[input]++ // Tambahkan jumlah
        suara partai pada map
    }
```

```

// Konversi map menjadi slice
for nama, suara := range NIM_2311102275_Map {
    NIM_2311102275_TabPartai =
append(NIM_2311102275_TabPartai, NIM_2311102275_Partai{Nama:
nama, Suara: suara})
}

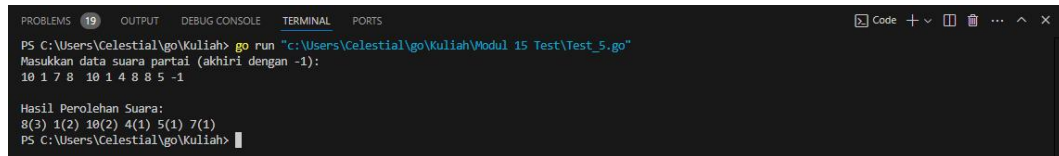
// Mengurutkan slice berdasarkan jumlah suara (descending)
sort.Slice(NIM_2311102275_TabPartai, func(i, j int) bool {
    if NIM_2311102275_TabPartai[i].Suara ==
NIM_2311102275_TabPartai[j].Suara {
        return NIM_2311102275_TabPartai[i].Nama <
NIM_2311102275_TabPartai[j].Nama // Jika suara sama, urutkan
berdasarkan nama partai
    }
    return NIM_2311102275_TabPartai[i].Suara >
NIM_2311102275_TabPartai[j].Suara
}))

// Menampilkan hasil
fmt.Println("\nHasil Perolehan Suara:")
var output []string
for _, partai := range NIM_2311102275_TabPartai {
    output = append(output, fmt.Sprintf("%d(%d)",
partai.Nama, partai.Suara))
}
fmt.Println(strings.Join(output, " "))
}

// Fungsi untuk mencari posisi partai dalam array
func NIM_2311102275_Posisi(tabPartai NIM_2311102275_TabPartai,
nama int) int {
    for i, partai := range tabPartai {
        if partai.Nama == nama { // Jika nama partai ditemukan
            return i
        }
    }
    return -1 // Jika tidak ditemukan
}

```

Screenshoot Output



The screenshot shows a VS Code terminal window with the following content:

```
PROBLEMS 19 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Celestial\go\Kuliah> go run "c:\Users\Celestial\go\Kuliah\Modul 15 Test\Test_5.go"
Masukkan data suara partai (akhiri dengan -1):
10 1 7 8 10 1 4 8 8 5 -1

Hasil Perolehan Suara:
8(3) 1(2) 10(2) 4(1) 5(1) 7(1)
PS C:\Users\Celestial\go\Kuliah>
```