# LAPORAN PRAKTIKUM
# ALGORITMA DAN PEMROGRAMAN 2

**UJIAN AKHIR**


**Disusun Oleh :**

**Haposan Felix Marcel Siregar / 2311102210**

**S1IF-11-06**

**Dosen Pengampu :**

**Abednego Dwi Septiadi**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

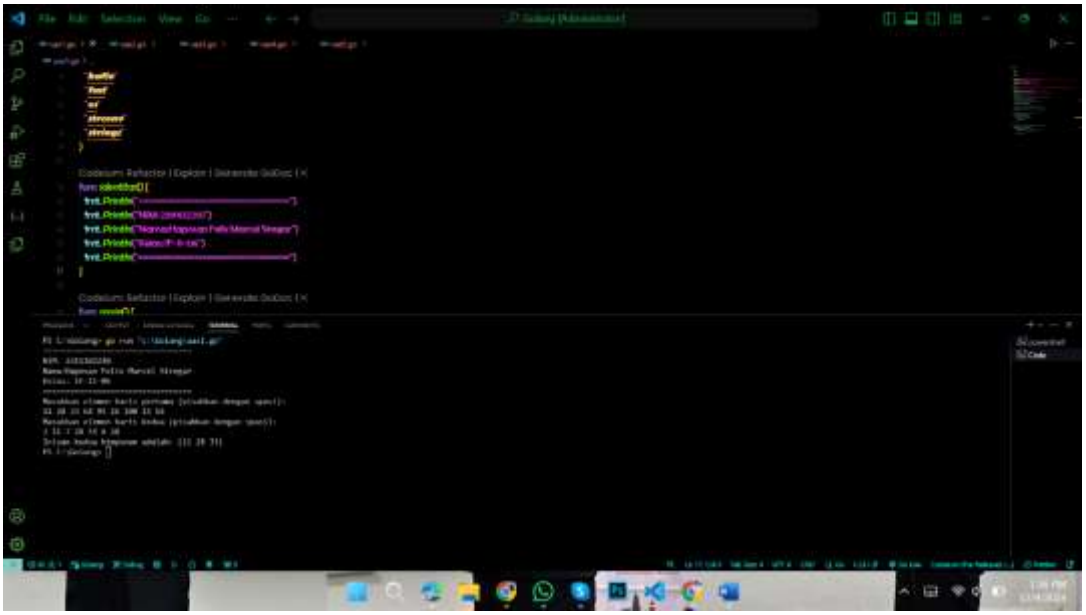

**2024**

1. **Code Program**

```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strconv"
    "strings"
)

func identitas() {
    fmt.Println("================================")
    fmt.Println("NIM: 2311102210")
    fmt.Println("Nama:Haposan Felix Marcel Siregar")
    fmt.Println("Kelas: IF-11-06")
    fmt.Println("================================")
}

func main() {
    identitas()
    //Fungsi ini  untuk membaca input dari user
    scanner := bufio.NewScanner(os.Stdin)
    fmt.Println("Masukkan elemen baris pertama (pisahkan dengan
spasi):")
    scanner.Scan()
    line1 := strings.Fields(scanner.Text())

    fmt.Println("Masukkan elemen baris kedua (pisahkan dengan
spasi):")
    scanner.Scan()
    line2 := strings.Fields(scanner.Text())

    // Konversi input ke integer dan buang elemen duplikat
    set1_2311102210 := toUniqueSet(line1)
    set2 := toUniqueSet(line2)

    // Cari irisan dari kedua himpunan
    intersection := findIntersection(set1_2311102210, set2)

    // Tampilkan hasil
    fmt.Println("Irisan kedua himpunan adalah:", intersection)
}

func toUniqueSet(elements []string) map[int]bool {
    set := make(map[int]bool)
    for _, el := range elements {
```

```
        num, err := strconv.Atoi(el)
        if err == nil {
            set[num] = true
        }
    }
    return set
}

func findIntersection(set1_2311102210, set2 map[int]bool) []int
{
    intersection := []int{}
    for num := range set1_2311102210 {
        if set2[num] {
            intersection = append(intersection, num)
        }
    }
    return intersection
}
```

**Output**

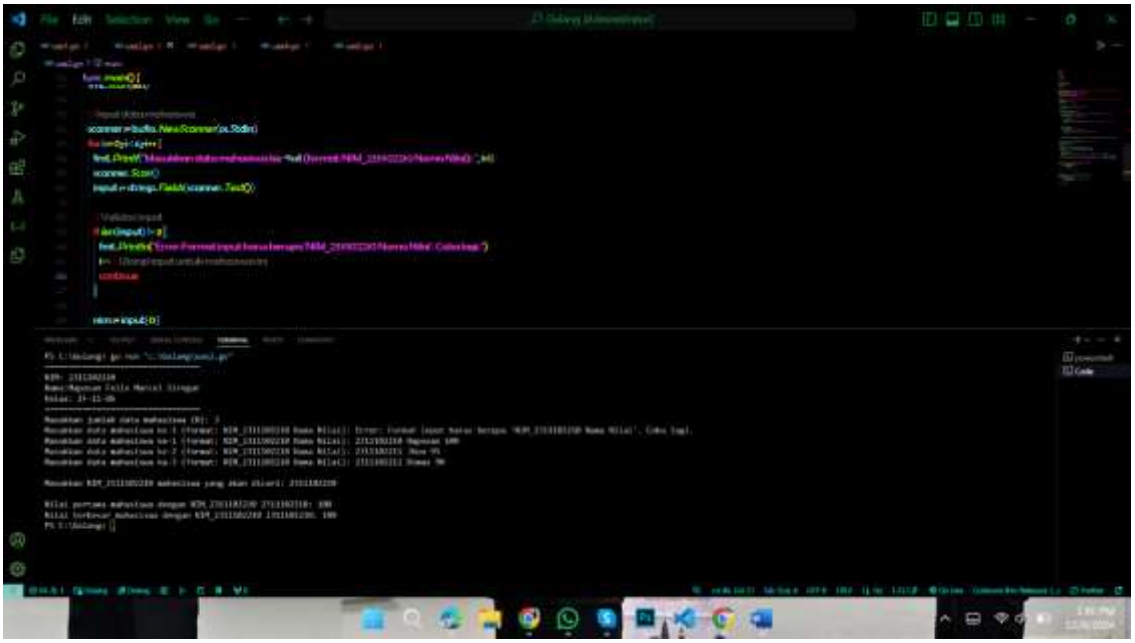


**2. Source Code**

```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strconv"
    "strings"
)

// Struct Mahasiswa
type Mahasiswa struct {
    NIM_2311102210   string
    Nama  string
    Nilai int
}

func identitas() {
    fmt.Println("================================")
    fmt.Println("NIM: 2311102210")
    fmt.Println("Nama:Haposan Felix Marcel Siregar")
    fmt.Println("Kelas: IF-11-06")
    fmt.Println("================================")
}

func main() {
    identitas()
    var mahasiswaList []Mahasiswa
    var n int

    // Input jumlah data mahasiswa
    fmt.Print("Masukkan jumlah data mahasiswa (N): ")
    fmt.Scan(&n)

    // Input data mahasiswa
    scanner := bufio.NewScanner(os.Stdin)
    for i := 0; i < n; i++ {
        fmt.Printf("Masukkan data mahasiswa ke-%d (format:
NIM_2311102210 Nama Nilai): ", i+1)
        scanner.Scan()
        input := strings.Fields(scanner.Text())

        // Validasi input
        if len(input) != 3 {
            fmt.Println("Error: Format input harus berupa
'NIM_2311102210 Nama Nilai'. Coba lagi.")
            i-- // Ulangi input untuk mahasiswa ini
            continue
        }
```

```
        nim := input[0]
        nama := input[1]
        nilai, err := strconv.Atoi(input[2])
        if err != nil {
            fmt.Println("Error: Nilai harus berupa angka. Coba
lagi.")
            i-- // Ulangi input untuk mahasiswa ini
            continue
        }

        mahasiswaList = append(mahasiswaList,
Mahasiswa{NIM_2311102210: nim, Nama: nama, Nilai: nilai})
    }

    // Input NIM_2311102210 yang akan dicari
    fmt.Print("\nMasukkan NIM_2311102210 mahasiswa yang akan
dicari: ")
    scanner.Scan()
    cariNIM := scanner.Text()

    // Pencarian nilai pertama dan nilai terbesar
    nilaiPertama := cariNilaiPertama(mahasiswaList, cariNIM)
    nilaiTerbesar := cariNilaiTerbesar(mahasiswaList, cariNIM)

    // Tampilkan hasil
    if nilaiPertama == -1 {
        fmt.Printf("\nMahasiswa dengan NIM_2311102210 %s tidak
ditemukan.\n", cariNIM)
    } else {
        fmt.Printf("\nNilai pertama mahasiswa dengan
NIM_2311102210 %s: %d\n", cariNIM, nilaiPertama)
        fmt.Printf("Nilai terbesar mahasiswa dengan
NIM_2311102210 %s: %d\n", cariNIM, nilaiTerbesar)
    }
}

// Fungsi mencari nilai pertama berdasarkan NIM_2311102210
func cariNilaiPertama(data []Mahasiswa, nim string) int {
    for _, m := range data {
        if m.NIM_2311102210 == nim {
            return m.Nilai
        }
    }
    return -1 // Jika tidak ditemukan
}

// Fungsi mencari nilai terbesar berdasarkan NIM_2311102210
func cariNilaiTerbesar(data []Mahasiswa, nim string) int {
    maxNilai := -1
    for _, m := range data {
```

**Output**



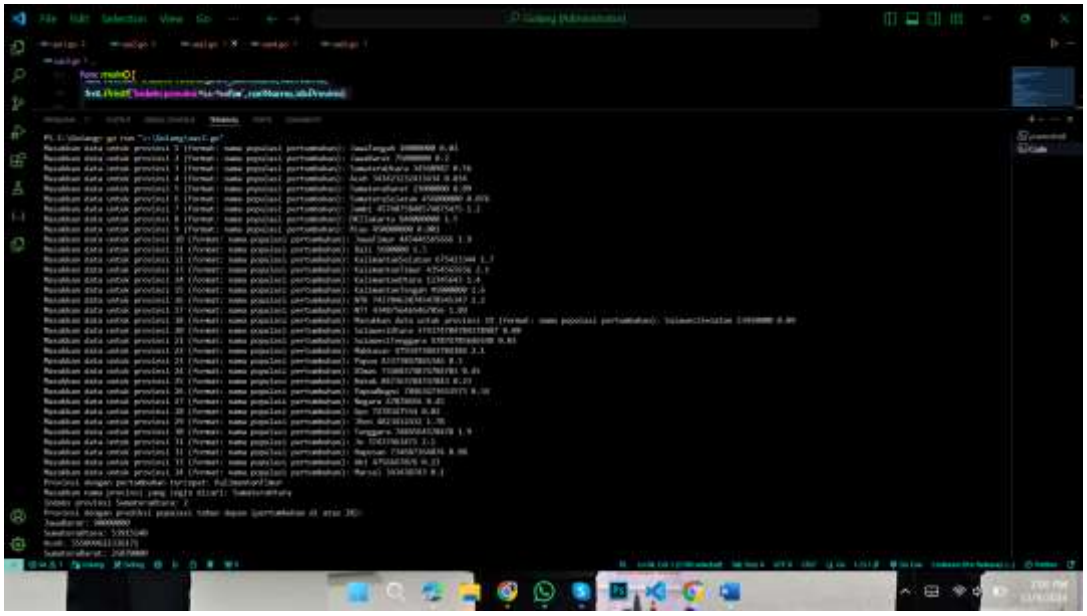
### 3. Source Code

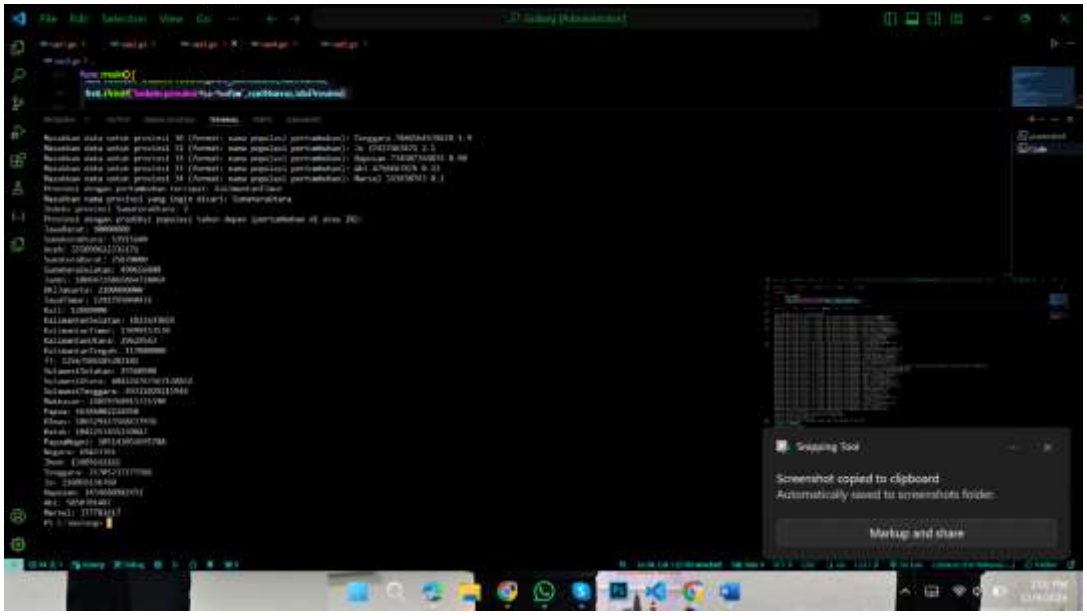
```
package main

import (
    "fmt"
    "strings"
)

const nProv = 34

type NamaProv [nProv]string
type PopProv [nProv]int
type TumbuhProv [nProv]float64

// Fungsi untuk input data
func InputData(prov *NamaProv, pop *PopProv, tumbuh *TumbuhProv)
{
    for i := 0; i < nProv; i++ {
        fmt.Printf("Masukkan data untuk provinsi %d (format: nama
populasi pertumbuhan): ", i+1)
        var nama string
        var populasi int
        var pertumbuhan float64
        fmt.Scan(&nama, &populasi, &pertumbuhan)
        prov[i] = nama
        pop[i] = populasi
        tumbuh[i] = pertumbuhan
    }
}
```

```
// Fungsi untuk mencari provinsi dengan pertumbuhan penduduk
tercepat
func ProvinsiTercepat(tumbuh TumbuhProv) int {
    maxIdx := 0
    for i := 1; i < nProv; i++ {
        if tumbuh[i] > tumbuh[maxIdx] {
            maxIdx = i
        }
    }
    return maxIdx
}

// Fungsi untuk menghitung prediksi jumlah penduduk
func Prediksi(prov NamaProv, pop PopProv, tumbuh TumbuhProv) {
    fmt.Println("Provinsi dengan prediksi populasi tahun depan
(pertumbuhan di atas 2%):")
    for i := 0; i < nProv; i++ {
        if tumbuh[i] > 0.02 {
            prediksiPop := float64(pop[i]) * (1 + tumbuh[i])
            fmt.Printf("%s: %.0f\n", prov[i], prediksiPop)
        }
    }
}

// Fungsi untuk mencari indeks provinsi berdasarkan nama
func IndeksProvinsi(prov NamaProv, nama string) int {
    for i := 0; i < nProv; i++ {
        if strings.EqualFold(prov[i], nama) {
            return i
        }
    }
    return -1
}

func main() {
    var prov_2311102210 NamaProv
    var pop PopProv
    var tumbuh TumbuhProv

    // Input data provinsi
    InputData(&prov_2311102210, &pop, &tumbuh)

    // Menentukan provinsi dengan pertumbuhan tercepat
    tercepatIdx := ProvinsiTercepat(tumbuh)
    fmt.Printf("Provinsi dengan pertumbuhan tercepat: %s\n",
prov_2311102210[tercepatIdx])

    // Input nama provinsi untuk dicari
    var cariNama string
    fmt.Print("Masukkan nama provinsi yang ingin dicari: ")
    fmt.Scan(&cariNama)

    // Mencari indeks provinsi berdasarkan nama
    idxProvinsi := IndeksProvinsi(prov_2311102210, cariNama)
    fmt.Printf("Indeks provinsi %s: %d\n", cariNama, idxProvinsi)
```

```
    // Menampilkan prediksi populasi provinsi dengan pertumbuhan
di atas 2%
    Prediksi(prov_2311102210, pop, tumbuh)
}
```

**Output**

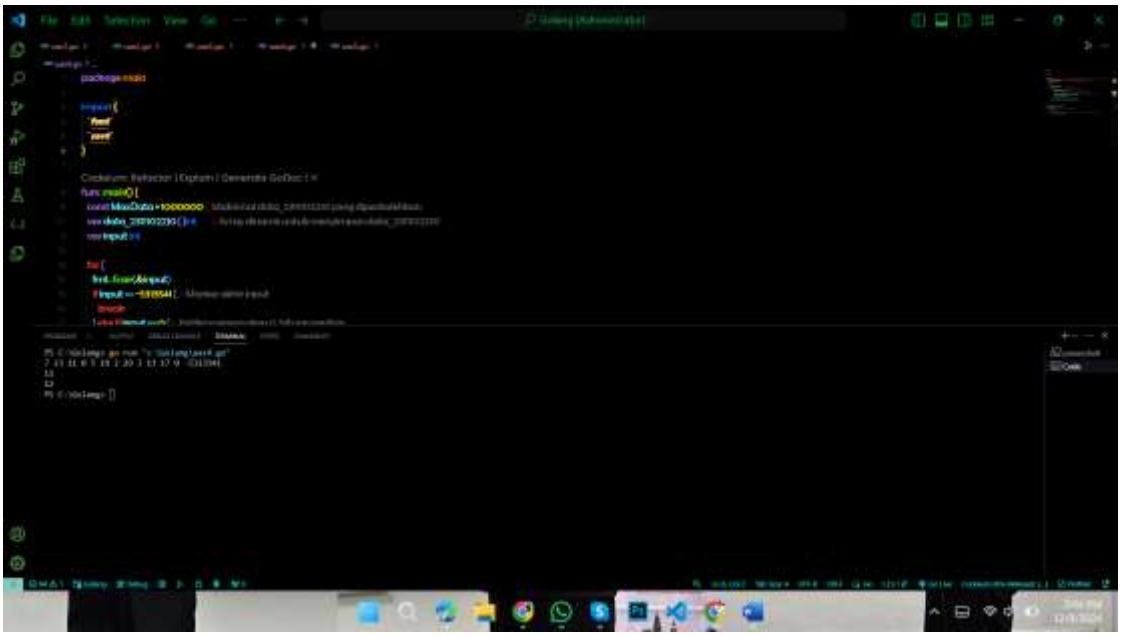




## 4. Source Code

```
package main

import (
    "fmt"
    "sort"
)

func main() {
    const MaxData = 1000000 // Maksimal data_2311102210 yang
diperbolehkan
    var data_2311102210 []int          // Array dinamis untuk
menyimpan data_2311102210
    var input int

    for {
        fmt.Scan(&input)
        if input == -5313541 { // Marker akhir input
            break
        } else if input == 0 { // Ketika menemukan 0, hitung
median
            if len(data_2311102210) == 0 {
                fmt.Println("Tidak ada data_2311102210 untuk
dihitung median.")
            } else {
                sort.Ints(data_2311102210) // Urutkan
data_2311102210 menggunakan fungsi bawaan Go
                median := calculateMedian(data_2311102210)
                fmt.Println(median)
            }
        } else { // Tambahkan data_2311102210 ke array
            data_2311102210 = append(data_2311102210, input)
        }
    }
}

// Fungsi untuk menghitung median
func calculateMedian(arr []int) int {
    n := len(arr)
    if n%2 == 1 { // Jika jumlah data_2311102210 ganjil
        return arr[n/2]
    }
    // Jika jumlah data_2311102210 genap, hitung rerata dua
nilai tengah dan bulatkan ke bawah
    return (arr[n/2-1] + arr[n/2]) / 2
}
```

**Output**



## 5. Source Code

```
package main

import "fmt"

// Konstanta jumlah maksimal partai
const NMAX = 1000000

// Struktur partai
type Partai struct {
    Nama_2311102210  int // Nama_2311102210 partai
    Suara int // Jumlah suara yang diperoleh
}

// Tipe array partai
type tabPartai [NMAX]Partai

func identitas() {
    fmt.Println("================================")
    fmt.Println("NIM: 2311102210")
    fmt.Println("Nama:Haposan Felix Marcel Siregar")
    fmt.Println("Kelas: IF-11-06")
    fmt.Println("================================")
}
```

```
func main() {
    identitas()
    var p tabPartai // Array untuk menyimpan data partai
    var n int       // Jumlah partai unik
    var input int

    // Membaca input suara secara berulang hingga menemukan -1
    for {
        fmt.Scan(&input)
        if input == -1 { // Marker untuk mengakhiri input
            break
        }

        // Mencari posisi partai dalam array
        pos := posisi(p, n, input)
        if pos == -1 { // Jika partai belum ada, tambahkan ke
array
            p[n] = Partai{Nama_2311102210: input, Suara: 1}
            n++
        } else { // Jika partai sudah ada, tambahkan suaranya
            p[pos].Suara++
        }
    }

    // Mengurutkan array p dengan insertion sort secara
descending
    insertionSort(&p, n)

    // Menampilkan hasil
    for i := 0; i < n; i++ {
        fmt.Printf("%d(%d) ", p[i].Nama_2311102210, p[i].Suara)
    }
    fmt.Println()
}

// Fungsi untuk mencari posisi partai dalam array
func posisi(t tabPartai, n int, nama int) int {
    for i := 0; i < n; i++ {
        if t[i].Nama_2311102210 == nama {
            return i // Mengembalikan indeks partai jika
ditemukan
        }
    }
    return -1 // Mengembalikan -1 jika partai tidak ditemukan
}

// Fungsi untuk mengurutkan array partai secara descending
berdasarkan jumlah suara
func insertionSort(t *tabPartai, n int) {
    for i := 1; i < n; i++ {
        temp := t[i]
        j := i - 1

        // Geser elemen yang lebih kecil
        for j >= 0 && (t[j].Suara < temp.Suara || (t[j].Suara
== temp.Suara && t[j].Nama_2311102210 > temp.Nama_2311102210))
```

```
{
            t[j+1] = t[j]
            j--
        }

        t[j+1] = temp
    }
}
```

**Output**