MODUL 15 – TEST

UJIAN AKHIR

ALGORITMA PEMROGRAMAN 2

Oleh :

Geranada Saputra Priambudi

2311102008

IF-11-06

Dosen Pengampu :

Abednego Dwi Septiadi, S.Kom., M.Kom.

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

I. UJIAN PRAKTIKUM

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.

   Source Code:

```
// Geranada Saputra Priambudi
// 2311102008
// IF-11-06

package main

import "fmt"

// Definisikan tipe data set sebagai array dengan ukuran max 2022
type set [2022]int

// Fungsi untuk memeriksa apakah suatu nilai ada dalam set
func exist(T set, n_2311102008 int, val int) bool {
    var i int = 0
    var status bool = false
    for i < n_2311102008 && !status {
        status = T[i] == val
        i++
    }
    return status
}

// Fungsi untuk memasukkan data ke dalam set
func inputSet(T *set, n_2311102008 *int) {
    *n_2311102008 = 0
    var bilangan int
    fmt.Scan(&bilangan)
      for  *n_2311102008  <  2022  &&  !exist(*T,  *n_2311102008,
bilangan) {
        T[*n_2311102008] = bilangan
        (*n_2311102008)++
        fmt.Scan(&bilangan)
    }
}

// Fungsi untuk mencari irisan dari dua himpunan
func findIntersection(T1, T2 set, n_2311102008, m int, T3 *set, h
*int) {
    var j int = 0
    *h = 0
    for j < n_2311102008 {
        if exist(T2, m, T1[j]) {
```

```
            T3[*h] = T1[j]
            (*h)++
        }
        j++
    }
}

// Fungsi untuk mencetak set
func printSet(T set, n_2311102008 int) {
    for i := 0; i < n_2311102008; i++ {
        fmt.Print(T[i], " ")
    }
    fmt.Println()
}

func main() {
    var s1, s2, s3 set
    var n1, n2, n3 int

    // Input set pertama
    inputSet(&s1, &n1)
    // Input set kedua
    inputSet(&s2, &n2)

    // Mencari irisan kedua set
    findIntersection(s1, s2, n1, n2, &s3, &n3)

    // Menampilkan hasil irisan
    printSet(s3, n3)
}
```

Output:

```
PS C:\Users\ACER\Documents\Alpro2\UjianPraktikum> go run "c:\Use
11 28 33 64 95 16 100 15 64
3 11 7 28 33 6 28
11 28 33
PS C:\Users\ACER\Documents\Alpro2\UjianPraktikum> go run "c:\Use
1 1
1 1
1
PS C:\Users\ACER\Documents\Alpro2\UjianPraktikum> go run "c:\Use
1 2 3 4 3
9 8 7 9
```

Penjelasan Program :

Program di atas adalah implementasi dalam bahasa Go yang digunakan untuk mencari irisan antara dua himpunan bilangan. Input terdiri dari dua baris, masing-masing merepresentasikan himpunan dengan elemen unik tanpa duplikasi. Fungsi exist digunakan untuk memeriksa keberadaan elemen dalam himpunan, sedangkan fungsi inputSet membaca input pengguna hingga ditemukan elemen yang duplikat. Fungsi findIntersection kemudian membandingkan kedua himpunan untuk menemukan elemen yang sama (irisan), yang hasilnya disimpan dalam himpunan

ketiga. Akhirnya, fungsi printSet digunakan untuk mencetak elemen dari hasil irisan tersebut ke layar.

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. Contoh isi tabel sebagai berikut:

114, Nana, 97	113, Jojo, 70	118, Rere, 88	116, Koko, 40	117, Keke, 90	116, Koko, 60	113, Jojo, 50	113, Jojo, 80	118, Rere, 88	119, Roro, 100

Pada contoh di atas, data Jojo ada tiga dengan 70 sebagai nilai pertama, kemudian 50, dan 80 sebagai nilai terakhir.
Definiskan struct dan array berikut:

```
constant nMax: integer = 51
type mahasiswa <NIM: string, nama:string, nilai:integer>
type arrayMahasiswa: array [1..nMax] of mahasiswa
```

Source Code:

```
// Geranada Saputra Priambudi
// 2311102008
// IF-11-06

package main

import "fmt"

const nMax int = 51

// Struct untuk mendefinisikan data mahasiswa
type mahasiswa struct {
    NIM_2311102008 string
    nama           string
    nilai          int
}

// Tipe data array untuk menyimpan data mahasiswa
type arrayMahasiswa [nMax]mahasiswa

// Fungsi untuk memasukkan data mahasiswa
func inputMhs(T *arrayMahasiswa, N *int) {
    fmt.Print("Masukkan jumlah mahasiswa: ")
    fmt.Scan(N)

    var i int = 0
    for i < *N {
```

```
        fmt.Printf("Masukkan NIM, Nama, dan Nilai mahasiswa ke-
%d: ", i+1)
        fmt.Scan(&T[i].NIM_2311102008, &T[i].nama, &T[i].nilai)
        i++
    }
}

// Fungsi untuk mencari nilai pertama berdasarkan NIM
func findFirstScore(T arrayMahasiswa, N int, nim string) int {
    var i int
    for i = 0; i < N; i++ {
        if T[i].NIM_2311102008 == nim {
           return i // Mengembalikan indeks pertama kali ditemukan
        }
    }
    return -1 // Mengembalikan -1 jika NIM tidak ditemukan
}

// Fungsi untuk mencari nilai terbesar berdasarkan NIM
func findMaxScore(T arrayMahasiswa, N int, nim string) int {
    found := findFirstScore(T, N, nim)
    if found != -1 {
        idxMax := found
        for i := found + 1; i < N; i++ {
                if T[i].NIM_2311102008 == nim && T[i].nilai >
T[idxMax].nilai {
                idxMax = i
            }
        }
        return idxMax
    }
    return found
}

func main() {
    var A arrayMahasiswa
    var M int
    var NIM_2311102008 string

    // Input data mahasiswa
    inputMhs(&A, &M)

    // Mencari NIM yang diminta
    fmt.Print("Masukkan NIM yang ingin dicari: ")
    fmt.Scan(&NIM_2311102008)

    // Mencari nilai pertama dan nilai terbesar
    idx1 := findFirstScore(A, M, NIM_2311102008)
    if idx1 == -1 {
        fmt.Println("NIM", NIM_2311102008, "tidak ditemukan.")
    } else {
        idx2 := findMaxScore(A, M, NIM_2311102008)
```

```
          fmt.Println("Nilai pertama dari NIM", NIM_2311102008,
"adalah", A[idx1].nilai)
         fmt.Println("Nilai terbesar dari NIM", NIM_2311102008,
"adalah", A[idx2].nilai)
    }
}
```

Output:

```
Masukkan jumlah mahasiswa: 10
Masukkan NIM, Nama, dan Nilai mahasiswa ke-1: 114 Nana 97
Masukkan NIM, Nama, dan Nilai mahasiswa ke-2: 113 Jojo 70
Masukkan NIM, Nama, dan Nilai mahasiswa ke-3: 118 Rere 88
Masukkan NIM, Nama, dan Nilai mahasiswa ke-4: 116 Koko 40
Masukkan NIM, Nama, dan Nilai mahasiswa ke-5: 117 Keke 90
Masukkan NIM, Nama, dan Nilai mahasiswa ke-6: 116 Koko 60
Masukkan NIM, Nama, dan Nilai mahasiswa ke-7: 113 Jojo 50
Masukkan NIM, Nama, dan Nilai mahasiswa ke-8: 113 Jojo 80
Masukkan NIM, Nama, dan Nilai mahasiswa ke-9: 118 Rere 88
Masukkan NIM, Nama, dan Nilai mahasiswa ke-10: 119 Roro 100
Masukkan NIM yang ingin dicari: 113
Nilai pertama dari NIM 113 adalah 70
Nilai terbesar dari NIM 113 adalah 80
```

Penjelasan Program :

Program di atas adalah implementasi dalam bahasa Go untuk mencatat data mahasiswa berupa NIM, nama, dan nilai menggunakan array dan struct. Data mahasiswa dimasukkan ke dalam array secara berurutan berdasarkan jumlah data yang sudah ada. Fungsi findFirstScore digunakan untuk mencari nilai pertama yang tercatat berdasarkan NIM, sedangkan fungsi findMaxScore digunakan untuk mencari nilai terbesar dari semua data yang terkait dengan NIM tersebut. Jika NIM tidak ditemukan, program akan memberi pesan bahwa data tidak ada.

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 (riil) 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%.

   Source Code:

```
// Geranada Saputra Priambudi
// 2311102008
// IF-11-06
```

```
package main

import "fmt"

const nProv int = 34

// Tipe data array untuk menyimpan nama provinsi, populasi, dan
angka pertumbuhan
type NamaProv_2311102008 [nProv]string
type PopProv [nProv]int
type TumbuhProv [nProv]float64

// Fungsi untuk memasukkan data provinsi, populasi, dan angka
pertumbuhan
func InputData(prov *NamaProv_2311102008, pop *PopProv, tumbuh
*TumbuhProv) {
    fmt.Println("Masukkan 34 nama provinsi, populasi, dan angka
pertumbuhan:")
    var i int
    for i = 0; i < nProv; i++ {
        fmt.Scan(&prov[i], &pop[i], &tumbuh[i])
    }
}

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

// Fungsi untuk menampilkan prediksi jumlah penduduk provinsi
dengan pertumbuhan > 2%
func Prediksi(prov NamaProv_2311102008, pop PopProv, tumbuh
TumbuhProv) {
     fmt.Println("Prediksi jumlah penduduk tahun depan untuk
provinsi dengan pertumbuhan di atas 2%:")
    var i int
    var result float64
    for i = 0; i < nProv; i++ {
        if tumbuh[i] > 0.02 {
            result = (1 + tumbuh[i]) * float64(pop[i])
             fmt.Printf("%s %.0f\n", prov[i], result) // Format
lebih rapi
        }
    }
}
```

```
// Fungsi untuk mencari indeks provinsi berdasarkan nama
func IndeksProvinsi(prov NamaProv_2311102008, nama string) int {
    var found int = -1
    var i int = 0
    for i < nProv && found == -1 {
        if prov[i] == nama {
            found = i
        }
        i++
    }
    return found
}

func main() {
    var TProvinsi NamaProv_2311102008
    var TPopulasi PopProv
    var TPertumbuhan TumbuhProv
    var cari string
    var idxTercepat, idxProvinsi int

    // Memasukkan data provinsi
    InputData(&TProvinsi, &TPopulasi, &TPertumbuhan)

    // Menerima nama provinsi yang ingin dicari
    fmt.Println("Masukkan nama provinsi yang ingin dicari:")
    fmt.Scan(&cari)

      // Mencari provinsi dengan pertumbuhan tercepat dan
menampilkan nama provinsi
    idxTercepat = ProvinsiTercepat(TPertumbuhan)
    fmt.Printf("Nama provinsi dengan angka pertumbuhan tercepat:
%s\n", TProvinsi[idxTercepat])

    // Mencari indeks provinsi yang dicari dan menampilkan nama
provinsi
    idxProvinsi = IndeksProvinsi(TProvinsi, cari)
    if idxProvinsi != -1 {
          fmt.Printf("Indeks provinsi yang dicari sesuai nama:
%d\n", idxProvinsi)
    } else {
        fmt.Println("Provinsi tidak ditemukan.")
    }

     // Menampilkan prediksi jumlah penduduk provinsi dengan
pertumbuhan lebih dari 2%
    Prediksi(TProvinsi, TPopulasi, TPertumbuhan)
}
```

Output:

```
PS C:\Users\ACER\Documents\Alpro2\UjianPraktikum> go run "c:\Users\ACER\Documents\Alpro2\UjianPraktikum\no3\soal3.go"
Masukkan 34 nama provinsi, populasi, dan angka pertumbuhan:
Jakarta 10000000 0.015
JawaBarat 50000000 0.03
JawaTengah 40000000 0.025
JawaTimur 39000000 0.018
SumateraUtara 13000000 0.02
Riau 6000000 0.03
Lampung 7000000 0.025
KalimantanBarat 4000000 0.018
KalimantanTimur 3500000 0.02
KalimantanSelatan 3300000 0.015
SulawesiSelatan 8000000 0.022
SulawesiUtara 2500000 0.017
SulawesiTenggara 3000000 0.018
Bali 4300000 0.02
Papua 3000000 0.035
PapuaBarat 900000 0.03
Maluku 1500000 0.02
MalukuUtara 1100000 0.018
NTT 5000000 0.02
NTB 4900000 0.025
Gorontalo 1200000 0.015
Banten 12000000 0.033
Yogyakarta 3500000 0.015
Aceh 5200000 0.017
SumateraBarat 7000000 0.018
SumateraSelatan 8500000 0.021
Jambi 3500000 0.02
Bengkulu 1800000 0.016
BangkaBelitung 1400000 0.02
KalimantanUtara 700000 0.025
SulawesiTengah 3000000 0.019
SulawesiBarat 1200000 0.015
KepulauanRiau 2000000 0.03
Sumba 1000000 0.01
Masukkan nama provinsi yang ingin dicari:
JawaTengah
Nama provinsi dengan angka pertumbuhan tercepat: Papua
Indeks provinsi yang dicari sesuai nama: 2
Prediksi jumlah penduduk tahun depan untuk provinsi dengan pertumbuhan di atas 2%:
JawaBarat 51500000
JawaTengah 41000000
Riau 6180000
Lampung 7175000
SulawesiSelatan 8176000
Papua 3105000
PapuaBarat 927000
NTB 5022500
Banten 12396000
SumateraSelatan 8678500
KalimantanUtara 717500
KepulauanRiau 2060000
```

Penjelasan Program :

Program di atas digunakan untuk mengolah data provinsi di Indonesia berdasarkan nama, jumlah populasi, dan angka pertumbuhan penduduk tahun 2018. Program menerima 34 data provinsi sebagai input, lalu menentukan provinsi dengan angka pertumbuhan penduduk tercepat menggunakan fungsi ProvinsiTercepat. Fungsi IndeksProvinsi mencari indeks suatu provinsi berdasarkan nama yang dimasukkan pengguna, sementara fungsi Prediksi menghitung dan menampilkan prediksi jumlah penduduk di tahun berikutnya untuk provinsi dengan pertumbuhan di atas 2%.

4. Kompetisi pemrograman yang baru saja berlalu diikuti oleh 17 tim dari berbagai perguruan tinggi ternama. Dalam kompetisi tersebut, setiap tim berlomba untuk menyelesaikan sebanyak mungkin problem yang diberikan. Dari 13 problem yang

diberikan, ada satu problem yang menarik. Problem tersebut mudah dipahami, hampir semua tim mencoba untuk menyelesaikannya, tetapi hanya 3 tim yang berhasil. Apa sih problemnya?
"Median adalah nilai tengah dari suatu koleksi data yang sudah terurut. Jika jumlah data genap, maka nilai median adalah rerata dari kedua nilai tengahnya. Pada problem ini, semua data merupakan bilangan bulat positif, dan karenanya rerata nilai tengah dibulatkan ke bawah."
Buatlah program median yang mencetak nilai median terhadap seluruh data yang sudah terbaca, jika data yang dibaca saat itu adalah 0. Masukan berbentuk rangkaian bilangan bulat.
Masukan berbentuk rangkaian bilangan bulat. Masukan tidak akan berisi lebih dari 1000000 data, tidak termasuk bilangan 0. Data 0 merupakan tanda bahwa median harus dicetak, tidak termasuk data yang dicari 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 ke dalam array.
b. dan setiap kali menemukan bilangan 0, urutkanlah data yang sudah tersimpan dengan menggunakan metoda selection sort dan ambil mediannya

Source Code:

```
// Geranada Saputra Priambudi
// 2311102008
// IF-11-06

package main

import "fmt"

// NMAX = batas maksimum jumlah data yang dapat disimpan
const NMAX = 1000000

// arrInt = tipe array yang berisi NMAX bilangan bulat
type arrInt [NMAX]int

// Fungsi sorting
// I.S. T adalah array dengan n elemen yang belum terurut
// F.S. T menjadi array dengan elemen-elemen terurut secara menaik
menggunakan algoritma selection sort
func sorting(T *arrInt, n int) {
    var pass_2311102008, idx_min, i, temp int
    // Iterasi untuk setiap elemen array
       for  pass_2311102008  =  1;  pass_2311102008  <=  n-1;
pass_2311102008++ {
        idx_min = pass_2311102008 - 1
        for i = pass_2311102008; i <= n-1; i++ {
            if T[idx_min] > T[i] {
                idx_min = i
            }
        }
```

```
        temp = T[idx_min]
        T[idx_min] = T[pass_2311102008-1]
        T[pass_2311102008-1] = temp
    }
}

// Fungsi median
// Mengembalikan median dari array T yang sudah terurut berisi n
elemen
func median(T arrInt, n int) float64 {
    var mid int = n / 2 // Indeks tengah array
    if n%2 == 0 {
        // Jika jumlah elemen genap, median adalah rata-rata dua
elemen tengah
        return float64(T[mid-1]+T[mid]) / 2.0
    } else {
        // Jika jumlah elemen ganjil, median adalah elemen tengah
        return float64(T[mid])
    }
}

// Fungsi utama (main)
func main() {
    var A arrInt
    var x, n int
    n = 0
    fmt.Scan(&x)

    // Proses setiap bilangan sampai menemukan marker -5313541
atau mencapai batas NMAX
    for x != -5313541 && n < NMAX {
        if x == 0 {
            sorting(&A, n)
            fmt.Println(median(A, n))
        } else {
            // Tambahkan bilangan ke array A
            A[n] = x
            n++
        }
        // Baca bilangan berikutnya
        fmt.Scan(&x)
    }
}
```

Output:

```
PS C:\Users\ACER\Documents\Alpro2\UjianPraktikum> go run "c:\U
7 23 11 0 5 19 2 29 3 13 17 0 -5313
11
12
```

Penjelasan Program :

Program diatas untuk membaca masukan berupa bilangan bulat dan menghitung nilai median setiap kali bilangan 0 ditemukan dalam alur masukan. Data yang dibaca akan disimpan dalam array hingga bilangan 0 ditemukan, lalu array tersebut akan diurutkan dengan metode *selection sort* dan median dihitung. Jika jumlah data genap, median adalah rata-rata dua elemen tengah yang dibulatkan ke bawah; jika jumlah data ganjil, median adalah elemen tengah dari data yang sudah diurutkan. Proses ini berakhir ketika bilangan akhir -5313541 ditemukan. Program mampu mengolah hingga satu juta bilangan, kecuali bilangan penanda seperti 0 dan -5313541.

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.

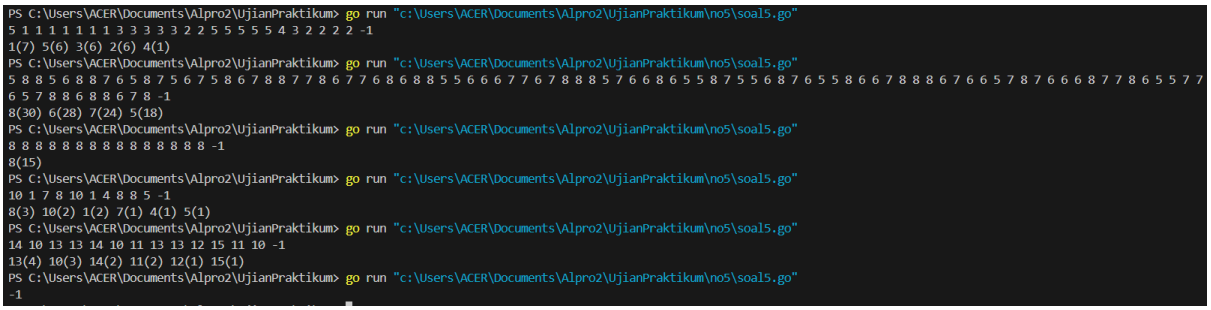
   Petunjuk : gunakan struct partai yang berisi nama dan suara. Data perolehan suara disimpan pada array of partai (kapasitas 1000000). Array tersebutlah yang diurutkan.

   Source Code:

```
// Geranada Saputra Priambudi
// 2311102008
// IF-11-06
package main

import "fmt"

const NMAX = 1000000

// struct partai
type partai struct {
    nama, suara int
}

// tipe tabPartai: array of partai dengan kapasitas NMAX
type tabPartai [NMAX]partai

func main() {
    // deklarasi variabel
    var p_2311102008 tabPartai
    var n, x, idx int
```

```
    // lakukan proses input suara secara berulang di sini, simpan
kedalam array p_2311102008, sehingga terdapat array p_2311102008
yang berisi hasil peroleh suara n partai.
    n = 0
    fmt.Scan(&x)
    for x != -1 {
        idx = posisi(p_2311102008, n, x)
        if idx == -1 {
            p_2311102008[n].nama = x
            p_2311102008[n].suara = 1
            n++
        } else {
            p_2311102008[idx].suara++
        }
        fmt.Scan(&x)
    }
    // lakukan proses pengurutan dengan insertion sort descending
berdasarkan jumlah suara yang diperoleh
    var pass, k int
    var temp partai
    for pass = 1; pass <= n-1; pass++ {
        k = pass
        temp = p_2311102008[k]
        for k > 0 && temp.suara > p_2311102008[k-1].suara {
            p_2311102008[k] = p_2311102008[k-1]
            k--
        }
        p_2311102008[k] = temp
    }
    // tampilkan array p_2311102008
    for k = 0; k < n; k++ {
                fmt.Printf("%v(%v)    ",    p_2311102008[k].nama,
p_2311102008[k].suara)
    }
}
func posisi(t tabPartai, n int, nama int) int {
    /* mengembalikan indeks partai yang memiliki nama yang dicari
pada array
       t yang berisi n partai atau -1 apabila tidak ditemukan ,
gunakan
       pencarian sekuensial */
    var i, ketemu int
    i = 0
    ketemu = -1
    for i < n && ketemu == -1 {
        if t[i].nama == nama {
            ketemu = i
        }
        i++
    }
    return ketemu
}
```

Output:



Penjelasan Program :

Program diatas digunakan untuk menghitung perolehan suara partai politik dalam pemilihan umum berdasarkan masukan dari pengguna. Setiap bilangan mewakili nama partai, dan setiap bilangan yang sama menunjukkan tambahan suara untuk partai tersebut. Input berakhir dengan nilai -1. Program menyimpan data partai dan jumlah suaranya menggunakan struktur partai dan array yang berkapasitas hingga satu juta elemen. Setelah semua suara dicatat, program mengurutkan data berdasarkan jumlah suara dengan metode *insertion sort* dalam urutan menurun (descending). Akhirnya, program mencetak daftar nama partai beserta jumlah suaranya dalam format partai(suara). Pencarian partai dilakukan dengan metode *sequential search* untuk memeriksa apakah partai sudah ada dalam daftar.