LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN II

MODUL XI

“PENGURUTAN DATA”

Oleh:

ZAHRINA ANTIKA MALAHATI

2311102109

IF 11 02

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Pengurutan merupakan hal yang tidak bisa dipisahkan dari dunia komputer. Adanya kebutuhan terhadap proses pengurutan memunculkan bermacam-macam metode pengurutan yang bertujuan untuk memperoleh metode pengurutan yang optimal. Dengan menggunakan algoritma yang baik dapat menghasilkan program yang efisien dari segi waktu dan hasil yang dicapai. Dalam modul ini kita telah mempelajari terkait pengurutan algoritma dengan insertion sort dan selection sort.

- **Algoritma Selection Sort**

  Prinsipnya adalah dengan memilih elemen dengan nilai paling rendah, lalu menukarnya dengan elemen ke-i. Cara kerjanya adalah dengan memilih elemen dengan nilai paling kecil pada indeks berikutnya, lalu menukarnya dengan elemen acuan.

  Pengurutan secara seleksi ini idenya adalah mencari nilai ekstrim pada sekumpulan data, kemudian meletakan pada posisi yang seharusnya.

  Notasi algortima dari selection sort adalah

```
i ← 1
while i <= n-1 do
  idx_min ← i-1
  j ← i
  while j < n do
    if a[idx_min] > a[j] then
      idx_min ← j
    endif
    j ← j + 1
  endwhile
  t ← a[idx_min]
  a[idx_min] ← a[i-1]
  a[i-1] ← t
  i ← i + 1
endwhile
```

- **Algortima Insertion Sort**

  Prinsipnya adalah dengan menyisipkan elemen ke dalam posisi yang tepat. Cara kerjanya adalah dengan mengambil elemen satu per satu dari list yang belum diurutkan, lalu memasukkannya ke dalam posisi yang benar di array yang sudah diurutkan.

  Pengurutan secara insertion ini idenya adalah menyisipkan suatu nilai pada posisi yang seharusnya. Berbeda dengan pengurutan seleksi, yang mana pada pengurutan ini tidak dilakukan pencarian nilai ekstrim terlebih dahulu, cukup memilih suatu nilai tertentu kemudian mencari posisinya secara sequential search. Pada penjelasan berikut ini data akan

diurut mengecil (descending), dan dengan indeks kecil ada di "kiri" dan indeks besar ada di "kanan".

Notasi Algoritma dari insertion sort adalah

```
i ← 1
while i <= n-1 do
  j ← i
  temp ← a[j]
  while j > 0 and temp > a[j-1] do
    a[j] ← a[j-1]
    j ← j - 1
  endwhile
  a[j] ← temp
  i ← i + 1
endwhile
```

II. GUIDED

❖ Guided 1

Source code

```
package main

import "fmt"

// Fungsi untuk mengurutkan array menggunakan selection sort
func selectionSort(arr []int) {
    n := len(arr)
    for i := 0; i < n-1; i++ {
        maxIdx := i
        for j := i + 1; j < n; j++ {
           if arr[j] > arr[maxIdx] { // Cari elemen terbesar
                maxIdx = j
            }
        }
        arr[i], arr[maxIdx] = arr[maxIdx], arr[i] // Tukar
elemen
    }
}

func main() {
    var n int
    fmt.Print("Masukkan jumlah daerah (n): ")
    fmt.Scan(&n)

    if n <= 0 || n >= 1000 {
        fmt.Println("n harus lebih besar dari 0 dan kurang
dari 1000.")
        return
    }
```

```
    for i := 0; i < n; i++ {
        var m int
          fmt.Printf("Masukkan jumlah rumah kerabat untuk
daerah ke-%d: ", i+1)
        fmt.Scan(&m)

        if m <= 0 || m >= 1000000 {
              fmt.Println("m harus lebih besar dari 0 dan
kurang dari 1000000.")
            return
        }

        // Masukkan nomor rumah
        houses := make([]int, m)
          fmt.Printf("Masukkan nomor rumah kerabat untuk
daerah ke-%d: ", i+1)
        for j := 0; j < m; j++ {
            fmt.Scan(&houses[j])
        }

        // Urutkan dengan selection sort
        selectionSort(houses)

        // Cetak hasil
        fmt.Printf("Hasil urutan rumah untuk daerah ke-%d:
", i+1)
        for _, house := range houses {
            fmt.Printf("%d ", house)
        }
        fmt.Println()
    }
}
```

Output

```
PS C:\PRAKALPRO2\MODUL11> go run "c:\PRAKALPRO2\MODUL11\guided1.go"
Masukkan jumlah daerah (n): 2
Masukkan jumlah rumah kerabat untuk daerah ke-1: 3
Masukkan nomor rumah kerabat untuk daerah ke-1: 4 5 6
Hasil urutan rumah untuk daerah ke-1: 6 5 4
Masukkan jumlah rumah kerabat untuk daerah ke-2: 4
Masukkan nomor rumah kerabat untuk daerah ke-2: 7 6 8 9
Hasil urutan rumah untuk daerah ke-2: 9 8 7 6
PS C:\PRAKALPRO2\MODUL11>
```

Deskripsi

Program adalah program dalam bahasa Go untuk mengurutkan nomor rumah dari beberapa daerah. Caranya, program akan meminta kita memasukkan jumlah daerah dan jumlah rumah di setiap daerah. Kemudian, kita akan memasukkan nomor-nomor rumah tersebut. Program lalu akan mengurutkan nomor rumah dari yang terkecil hingga terbesar menggunakan metode *selection sort*. Hasil akhir berupa daftar nomor rumah yang sudah terurut untuk setiap daerah akan ditampilkan.

❖ **Guided 2**

Source code

```
package main

import (
    "fmt"
    "math"
)

// Fungsi insertion sort untuk mengurutkan arraya
func insertionSort(arr []int) {
    n := len(arr)
    for i := 1; i < n; 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 memeriksa apakah data berjarak tetap
func isDataConsistentlySpaced(arr []int) (bool, int) {
    if len(arr) < 2 {
         return true, 0 // Array dengan kurang dari 2
elemen dianggap berjarak tetap
    }

    // Hitung selisih awal
    diff := int(math.Abs(float64(arr[1] - arr[0])))

    for i := 1; i < len(arr)-1; i++ {
         currentDiff := int(math.Abs(float64(arr[i+1] -
arr[i])))
        if currentDiff != diff {
             return false, 0 // Jika ada selisih yang
berbeda, tidak berjarak tetap
        }
    }

    return true, diff
}

func main() {
    var data []int
    var input int

     fmt.Println("Masukkan data (akhiri dengan bilangan
negatif):")
```

```
    for {
        fmt.Scan(&input)
        if input < 0 {
            break
        }
        data = append(data, input)
    }

    // Urutkan data menggunakan insertion sort
    insertionSort(data)

    // Periksa apakah data berjarak tetap
    isConsistent, diff := isDataConsistentlySpaced(data)

    // Cetak hasil
    fmt.Println("Hasil pengurutan:", data)
    if isConsistent {
        fmt.Printf("Data berjarak %d\n", diff)
    } else {
        fmt.Println("Data berjarak tidak tetap")
    }
}
```

Output

```
PS C:\PRAKALPRO2\MODUL11> go run "c:\PRAKALPRO2\MODUL11\guided2.go"
Masukkan data (akhiri dengan bilangan negatif):
2 4 6 8 -10
Hasil pengurutan: [2 4 6 8]
Data berjarak 2
PS C:\PRAKALPRO2\MODUL11>
```

Deskripsi

Program adalah program dalam bahasa Go untuk mengurutkan sekumpulan angka yang diinputkan pengguna dan kemudian memeriksa apakah angka-angka tersebut memiliki jarak yang sama. Program ini menggunakan metode pengurutan *insertion sort* yang efisien untuk data berukuran kecil hingga sedang. Setelah data terurut, program akan menghitung selisih antara setiap angka yang berdekatan. Jika semua selisih tersebut sama, maka dapat disimpulkan bahwa data tersebut memiliki jarak yang konsisten. Dengan kata lain, program ini tidak hanya mengurutkan data, tetapi juga menganalisis pola tertentu dalam data tersebut. Hasil akhir berupa daftar angka yang telah terurut dan informasi mengenai apakah jarak antara angka-angka tersebut konsisten.

III. UNGUIDED

❖ Unguided 1

Soal

2) Belakangan diketahui ternyata Hercules itu tidak berani menyeberang jalan, maka selalu diusahakan agar hanya menyeberang jalan sesedikit mungkin, hanya diujung jalan. Karena nomor rumah sisi kiri jalan selalu ganjil dan sisi kanan jalan selalu genap, maka buatlah program **kerabat dekat** yang akan menampilkan nomor rumah mulai dari nomor yang ganjil lebih dulu terurut membesar dan kemudian menampilkan nomor rumah dengan nomor genap terurut mengecil.

Format **Masukan** masih persis sama seperti sebelumnya.

Keluaran terdiri dari n baris, yaitu rangkaian rumah kerabatnya terurut membesar untuk nomor ganjil, diikuti dengan terurut mengecil untuk nomor genap, di masing-masing daerah.

No	Masukan	Keluaran
1	3 5 2 1 7 9 13 6 189 15 27 39 75 133 3 4 9 1	1 13 12 8 2 15 27 39 75 133 189 8 4 2

Keterangan: Terdapat 3 daerah dalam contoh masukan. Baris kedua berisi campuran bilangan ganjil dan genap. Baris berikutnya hanya berisi bilangan ganjil, dan baris terakhir hanya berisi bilangan genap.

Petunjuk:

- Waktu pembacaan data, bilangan ganjil dan genap dipisahkan ke dalam dua array yang berbeda, untuk kemudian masing-masing diurutkan tersendiri.
- Atau, tetap disimpan dalam satu array, diurutkan secara keseluruhan. Tetapi pada waktu pencetakan, mulai dengan mencetak semua nilai ganjil lebih dulu, kemudian setelah selesai cetaklah semua nilai genapnya.

Source code

```
package main

import "fmt"

func selectionSort(arr []int, ascending bool) {
    n := len(arr)
    for i := 0; i < n-1; i++ {
        targetIdx := i
        for j := i + 1; j < n; j++ {
             if (ascending && arr[j] < arr[targetIdx]) ||
(!ascending && arr[j] > arr[targetIdx]) {
                targetIdx = j
            }
        }
        arr[i], arr[targetIdx] = arr[targetIdx], arr[i]
```

```
    }
}

func main() {
    var n int
    fmt.Print("Masukkan jumlah daerah (n): ")
    fmt.Scan(&n)

    if n <= 0 || n >= 1000 {
        fmt.Println("n harus lebih besar dari 0 dan kurang
dari 1000.")
        return
    }

    for i := 0; i < n; i++ {
        var m int
          fmt.Printf("Masukkan jumlah rumah kerabat untuk
daerah ke-%d: ", i+1)
        fmt.Scan(&m)

        if m <= 0 || m >= 1000000 {
              fmt.Println("m harus lebih besar dari 0 dan
kurang dari 1000000.")
            return
        }

        houses := make([]int, m)
          fmt.Printf("Masukkan nomor rumah kerabat untuk
daerah ke-%d: ", i+1)
        for j := 0; j < m; j++ {
            fmt.Scan(&houses[j])
        }

        var oddHouses []int
        var evenHouses []int

        for _, house := range houses {
            if house%2 == 1 {
                  oddHouses = append(oddHouses, house) //
Nomor ganjil
            } else {
                evenHouses = append(evenHouses, house) //
Nomor genap
            }
        }

        // Urutkan nomor rumah ganjil membesar
        selectionSort(oddHouses, true)

        // Urutkan nomor rumah genap mengecil
        selectionSort(evenHouses, false)

        // Cetak hasil
        fmt.Printf("Hasil urutan rumah untuk daerah ke-%d:
", i+1)
```

```
        // Cetak nomor rumah ganjil
        for _, house := range oddHouses {
            fmt.Printf("%d ", house)
        }

        // Cetak nomor rumah genap
        for _, house := range evenHouses {
            fmt.Printf("%d ", house)
        }

        fmt.Println()
    }
}

// Zahrina Antika Malahati_2311102109
```

Output

```
PS C:\PRAKALPRO2\MODUL11> go run "c:\PRAKALPRO2\MODUL11\unguided1.go"
Masukkan jumlah daerah (n): 3
Masukkan jumlah rumah kerabat untuk daerah ke-1: 2
Masukkan nomor rumah kerabat untuk daerah ke-1: 4 3
Hasil urutan rumah untuk daerah ke-1: 3 4
Masukkan jumlah rumah kerabat untuk daerah ke-2: 3
Masukkan nomor rumah kerabat untuk daerah ke-2: 5 7 6
Hasil urutan rumah untuk daerah ke-2: 5 7 6
Masukkan jumlah rumah kerabat untuk daerah ke-3: 5
Masukkan nomor rumah kerabat untuk daerah ke-3: 12 14 11 17 15
Hasil urutan rumah untuk daerah ke-3: 11 15 17 14 12
PS C:\PRAKALPRO2\MODUL11>
```

Deskripsi

Program ini adalah program dalam bahasa Go untuk mengurutkan nomor rumah kerabat berdasarkan daerah dan paritas (ganjil atau genap). Program ini meminta pengguna untuk memasukkan jumlah daerah dan jumlah rumah kerabat di setiap daerah. Kemudian, nomor rumah akan dibagi menjadi dua kelompok: nomor ganjil dan nomor genap. Masing-masing kelompok ini kemudian diurutkan menggunakan algoritma *selection sort*. Nomor rumah ganjil akan diurutkan dari yang terkecil ke yang terbesar, sedangkan nomor rumah genap akan diurutkan dari yang terbesar ke yang terkecil.

❖ Unguided 2

Soal

3) 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 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 -5313.

Keluaran adalah median yang diminta, satu data per baris.

No	Masukan	Keluaran
1	7 23 11 0 5 19 2 29 3 13 17 0 -5313	11 12

Keterangan:

Sampai bilangan 0 yang pertama, data terbaca adalah 7 23 11, setelah tersusun: 7 11 23, maka median saat itu adalah 11.
Sampai bilangan 0 yang kedua, data adalah 7 23 11 5 19 2 29 3 13 17, setelah tersusun diperoleh: 2 3 5 7 **11 13** 17 19 23 29. Karena ada 10 data, genap, maka median adalah (11+13)/2=12.

Petunjuk:

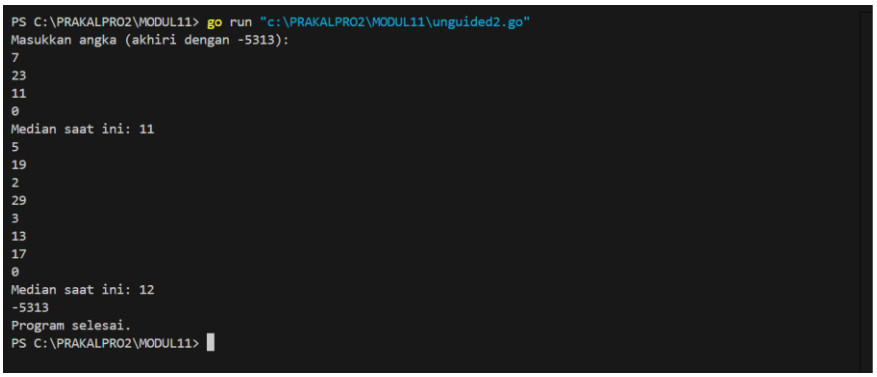
Untuk setiap data bukan 0 (dan bukan marker -5313541) simpan ke dalam array, Dan setiap kali menemukan bilangan 0, urutkanlah data yang sudah tersimpan dengan menggunakan metode insertion sort dan ambil mediannya.

Source Code

```
package main

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

func main() {
    var numbers []int
    scanner := bufio.NewScanner(os.Stdin)

    fmt.Println("Masukkan angka (akhiri dengan -5313):")
    for {
```

```
        scanner.Scan()
        input := scanner.Text()

        // Konversi input menjadi angka
        num, err := strconv.Atoi(input)
        if err != nil {
            fmt.Println("Masukkan angka valid.")
            continue
        }

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

        // Jika input adalah 0, hitung dan cetak median
        if num == 0 {
            if len(numbers) == 0 {
                    fmt.Println("Belum ada angka untuk
dihitung median.")
                continue
            }
            // Urutkan data
            sort.Ints(numbers)

            // Hitung median
            mid := len(numbers) / 2
            var median int
            if len(numbers)%2 == 0 {
                median = (numbers[mid-1] + numbers[mid])
/ 2
            } else {
                median = numbers[mid]
            }
            fmt.Println("Median saat ini:", median)
        } else {
            // Tambahkan angka ke dalam array
            numbers = append(numbers, num)
        }
    }

    fmt.Println("Program selesai.")
}

// Zahrina Antika Malahati_2311102109
```

Output



Deskripsi

Program ini adalah program dalam bahasa Go untuk menghitung median dari sekumpulan angka yang dimasukkan secara berurutan oleh pengguna. Pengguna dapat terus memasukkan angka hingga mereka memasukkan angka -5313 untuk mengakhiri input. Setelah itu, program akan mengurutkan semua angka yang telah dimasukkan, kemudian mencari nilai tengahnya (median). Jika jumlah angka genap, median adalah rata-rata dari dua angka tengah. Hasil perhitungan median akan ditampilkan setiap kali pengguna memasukkan angka 0. Program ini juga dilengkapi dengan fitur validasi input untuk memastikan hanya angka yang valid yang diproses.

❖ Unguided 3

Soal

2) Sebuah program perpustakaan digunakan untuk mengelola data buku di dalam suatu perpustakaan. Misalnya terdefinisi struct dan array seperti berikut ini:

```
const nMax : integer = 7919
type Buku = <
    id, judul, penulis, penerbit : string
    eksemplar, tahun, rating : integer >

type DaftarBuku = array [ 1..nMax] of Buku
Pustaka : DaftarBuku
nPustaka: integer
```

Masukan terdiri dari beberapa baris. Baris pertama adalah bilangan bulat N yang menyatakan banyaknya data buku yang ada di dalam perpustakaan. N baris berikutnya, masing-masingnya adalah data buku sesuai dengan atribut atau field pada struct. Baris terakhir adalah bilangan bulat yang menyatakan rating buku yang akan dicari.

Keluaran terdiri dari beberapa baris. Baris pertama adalah data buku terfavorit, baris kedua adalah lima judul buku dengan rating tertinggi, selanjutnya baris terakhir adalah data buku yang dicari sesuai rating yang diberikan pada masukan baris terakhir.

Lengkapi subprogram-subprogram dibawah ini, sesuai dengan I.S. dan F.S yang diberikan.

```
procedure DaftarkanBuku(in/out pustaka : DaftarBuku, n : integer)
{I.S. sejumlah n data buku telah siap para piranti masukan
 F.S. n berisi sebuah nilai, dan pustaka berisi sejumlah n data buku}

procedure CetakTerfavorit(in pustaka : DaftarBuku, in n : integer)
{I.S. array pustaka berisi n buah data buku dan belum terurut
 F.S. Tampilan data buku (judul, penulis, penerbit, tahun)
terfavorit, yaitu memiliki rating tertinggi}

procedure UrutBuku( in/out pustaka : DaftarBuku, n : integer )
{I.S. Array pustaka berisi n data buku
 F.S. Array pustaka terurut menurun/mengecil terhadap rating.
Catatan: Gunakan metoda Insertion sort}

procedure Cetak5Terbaru( in pustaka : DaftarBuku, n integer )
{I.S. pustaka berisi n data buku yang sudah terurut menurut rating
 F.S. Laporan 5 judul buku dengan rating tertinggi
 Catatan: Isi pustaka mungkin saja kurang dari 5}

procedure CariBuku(in pustaka : DaftarBuku, n : integer, r : integer )
{I.S. pustaka berisi n data buku yang sudah terurut menurut rating
 F.S. Laporan salah satu buku (judul, penulis, penerbit, tahun,
eksemplar, rating) dengan rating yang diberikan. Jika tidak ada buku
dengan rating yang ditanyakan, cukup tuliskan "Tidak ada buku dengan
rating seperti itu". Catatan: Gunakan pencarian biner/belah dua.}
```

Source Code

```
package main

import (
    "fmt"
    "sort"
)

type Buku struct {
    ID        int
```

```
    Judul     string
    Penulis   string
    Penerbit  string
    Eksemplar int
    Tahun     int
    Rating    int
}

type DaftarBuku struct {
    Pustaka  []Buku
    nPustaka int
}

// Fungsi untuk memasukkan daftar buku
func DaftarkanBuku(pustaka *DaftarBuku, n int) {
    pustaka.nPustaka = n
    pustaka.Pustaka = make([]Buku, n)

    for i := 0; i < n; i++ {
        fmt.Printf("Masukkan data buku ke-%d (ID, Judul,
Penulis, Penerbit, Eksemplar, Tahun, Rating):\n", i+1)
        fmt.Scanf("%d\n", &pustaka.Pustaka[i].ID)
        fmt.Scanln(&pustaka.Pustaka[i].Judul)
        fmt.Scanln(&pustaka.Pustaka[i].Penulis)
        fmt.Scanln(&pustaka.Pustaka[i].Penerbit)
                    fmt.Scanf("%d     %d     %d\n",
&pustaka.Pustaka[i].Eksemplar,
&pustaka.Pustaka[i].Tahun, &pustaka.Pustaka[i].Rating)
    }
}

// Fungsi untuk mencetak buku dengan rating tertinggi
func CetakTerfavorit(pustaka DaftarBuku) {
    var favorit Buku
    for _, buku := range pustaka.Pustaka {
        if buku.Rating > favorit.Rating {
            favorit = buku
        }
    }
    fmt.Println("Buku Terfavorit:")
     fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s,
Tahun:    %d\n",    favorit.Judul,    favorit.Penulis,
favorit.Penerbit, favorit.Tahun)
}

// Fungsi untuk mengurutkan buku berdasarkan rating
(descending)
func UrutBuku(pustaka *DaftarBuku) {
     sort.SliceStable(pustaka.Pustaka, func(i, j int)
bool {
                return   pustaka.Pustaka[i].Rating   >
pustaka.Pustaka[j].Rating
    })
}

// Fungsi untuk mencetak 5 buku dengan rating tertinggi
```

```
func CetakTerbaru(pustaka DaftarBuku) {
    fmt.Println("5 Buku dengan Rating Tertinggi:")
    for i := 0; i < 5 && i < pustaka.nPustaka; i++ {
        buku := pustaka.Pustaka[i]
       fmt.Printf("%d. Judul: %s, Penulis: %s, Penerbit:
%s, Tahun: %d\n",
           i+1, buku.Judul, buku.Penulis, buku.Penerbit,
buku.Tahun)
    }
}

// Fungsi untuk mencari buku berdasarkan rating tertentu
func CariBuku(pustaka DaftarBuku, rating int) {
    fmt.Printf("Buku dengan rating %d:\n", rating)
    found := false
    for _, buku := range pustaka.Pustaka {
        if buku.Rating == rating {
           fmt.Printf("Judul: %s, Penulis: %s, Penerbit:
%s, Tahun: %d\n",
                buku.Judul, buku.Penulis, buku.Penerbit,
buku.Tahun)
            found = true
        }
    }
    if !found {
           fmt.Println("Tidak  ada  buku  dengan  rating
tersebut.")
    }
}

func main() {
    var daftarBuku DaftarBuku
    var n, cariRating int

    // Input jumlah buku
    fmt.Println("Masukkan jumlah buku:")
    fmt.Scanf("%d\n", &n)

    // Daftarkan buku
    DaftarkanBuku(&daftarBuku, n)

    // Cetak buku terfavorit
    CetakTerfavorit(daftarBuku)

    // Urutkan buku berdasarkan rating
    UrutBuku(&daftarBuku)

    // Cetak 5 buku dengan rating tertinggi
    CetakTerbaru(daftarBuku)

    // Cari buku berdasarkan rating
    fmt.Println("Masukkan rating untuk mencari buku:")
    fmt.Scanf("%d\n", &cariRating)
    CariBuku(daftarBuku, cariRating)
}
```

```
// Zahrina Antika Malahati_2311102109
```

Output

Deskripsi

Program ini merupakan sebuah sistem perpustakaan sederhana yang dirancang untuk mengelola data buku. Program ini memungkinkan pengguna untuk memasukkan data buku secara manual, seperti judul, penulis, penerbit, dan rating. Setelah data buku tersimpan, program dapat digunakan untuk mencari buku dengan rating tertinggi, mengurutkan buku berdasarkan rating, menampilkan beberapa buku dengan rating terbaik, serta mencari buku berdasarkan rating tertentu. Program ini menggunakan struktur data untuk merepresentasikan buku dan koleksi buku, serta memanfaatkan fungsi pengurutan bawaan Go untuk mengurutkan data buku berdasarkan rating.