LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL XII & XIII

PENGURUTAN DATA

Disusun Oleh :

Raihan Ramadhan/2311102040

IF-11-05

Dosen Pengampu :

Arif Amrulloh, S.Kom., M.Kom

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Selection sort adalah salah satu algoritma pengurutan yang sederhana. Prinsip kerjanya adalah dengan mencari elemen terkecil dari bagian array yang belum terurut, lalu menukarnya dengan elemen pertama dari bagian tersebut. Proses ini diulang terus-menerus hingga seluruh array terurut.Selection sort masih efektif digunakan untuk mengurut kumpulan data yang relatif kecil. Algoritma ini mudah diimplementasikan dan dapat menjadi pilihan yang baik ketika membutuhkan solusi pengurutan yang sederhana.

Algoritma ini dikenal juga dengan nama ***Selection Sort***, yang mana pada algoritma ini melibatkan dua proses yaitu pencarian indeks nilai ekstrim dan proses pertukaran dua nilai atau *swap*.

	Notasi Algoritma	Notasi dalam bahasa Go
1	`i ← 1`	`i = 1`
2	`while i <= n-1 do`	`for i <= n-1 {`
3	`    idx_min ← i – 1`	`    idx_min = i – 1`
4	`    j ← i`	`    j = i`
5	`    while j < n do`	`    for j < n {`
6	`        if a[idx_min] > a[j] then`	`        if a[idx_min] > a[j] {`
7	`            idx_min ← j`	`            idx_min = j`
8	`        endif`	`        }`
9	`        j ← j + 1`	`        j = j + 1`
10	`    endwhile`	`    }`
11	`    t ← a[idx_min]`	`    t = a[idx_min]`
12	`    a[idx_min] ← a[i-1]`	`    a[idx_min] = a[i-1]`
13	`    a[i-1] ← t`	`    a[i-1] = t`
14	`    i ← i + 1`	`    i = i + 1`
15	`endwhile`	`}`

Insertion sort adalah salah satu algoritma pengurutan yang cukup sederhana. Prinsip kerjanya adalah dengan membagi array menjadi dua bagian,bagian yang sudah terurut dan bagian yang belum terurut. Elemen-elemen dari bagian yang belum terurut akan diambil satu per satu, lalu disisipkan ke dalam posisi yang tepat di bagian yang sudah terurut. Proses ini dilakukan dengan cara membandingkan elemen yang akan disisipkan dengan elemen-elemen di bagian yang sudah terurut, kemudian menggeser elemen yang lebih besar ke kanan untuk memberikan ruang bagi elemen yang akan disisipkan. Langkah-langkah ini terus diulang hingga semua elemen telah disisipkan ke dalam posisi yang benar, sehingga seluruh array menjadi terurut. Insertion sort dapat lebih efisien dalam kasus-kasus tertentu, terutama ketika array sudah hampir terurut.

Algoritma ini dikenal juga dengan nama ***Insertion Sort***, yang mana pada algoritma ini melibatkan dua proses yaitu pencarian sekuensial dan penyisipan.

	Notasi Algoritma	Notasi dalam bahasa Go
1	`i ← 1`	`i = 1`
2	`while i <= n-1 do`	`for i <= n-1 {`
3	`    j ← i`	`    j = i`
4	`    temp ← a[j]`	`    temp = a[j]`
5	`    while j > 0 and temp > a[j-1] do`	`    for j > 0 && temp > a[j-1] {`
6	`        a[j] ← a[j-1]`	`        a[j] = a[j-1]`
7	`        j ← j – 1`	`        j = j – 1`
8	`    endwhile`	`    }`
9	`    a[j] ← temp`	`    a[j] = temp`
10	`    i ← i + 1`	`    i = i + 1`
11	`endwhile`	`}`

II. Guided

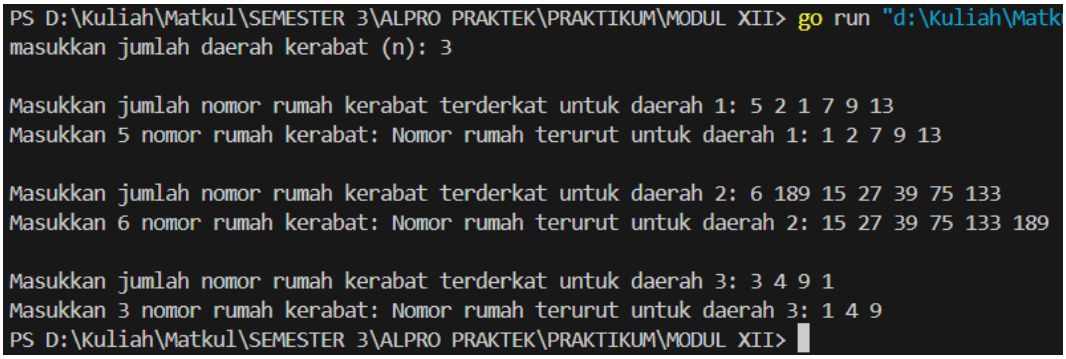
1. Guided 1

Source Code

```
package main

import "fmt"

func selectionSort(arr []int, n int) {
    for i := 0; i < n-1; i++ {
        idxMin := i
        for j := i + 1; j < n; j++ {
            //cari elemen terkecol
            if arr[j] < arr[idxMin] {
                idxMin = j
            }
        }
        //tukar elemen terkecil dengan elemen di posisi i
        arr[i], arr[idxMin] = arr[idxMin], arr[i]
    }
}

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

    //proses tiap daerah
    for daerah := 1; daerah <= n; daerah++ {
        var m int
        fmt.Printf("\nMasukkan jumlah nomor rumah kerabat terderkat
untuk daerah %d: ", daerah)
        fmt.Scan(&m)

        //membaca nomor rumah untuk daerah ini
        arr := make([]int, m)
        fmt.Printf("Masukkan %d nomor rumah kerabat: ", m)
        for i := 0; i < m; i++ {
            fmt.Scan(&arr[i])
        }
        //urutkan array dari terkecil ke terbesar
        selectionSort(arr, m)
```

```
        //tampillkan hasil
        fmt.Printf("Nomor rumah terurut untuk daerah %d: ", daerah)
        for _, num := range arr {
            fmt.Printf("%d ", num)
        }
        fmt.Println()

    }

}
```

Screenshot Output



Deskripsi Program

Program tersebut menggunakan algoritma selection sort untuk mengurutkan nomor rumah kerabat di setiap daerah. Pertama, pengguna diminta memasukkan jumlah daerah (n), kemudian untuk setiap daerah, mereka akan menginput jumlah nomor rumah (m) beserta daftar nomor rumahnya. Algoritma selection sort bekerja dengan menemukan elemen terkecil dalam array dan menukarnya dengan elemen di posisi tertentu, sehingga menghasilkan daftar nomor rumah yang terurut dari kecil ke besar. Setelah proses pengurutan selesai, hasilnya ditampilkan untuk masing-masing daerah. Program ini dirancang untuk memproses data secara terpisah per daerah, sehingga mempermudah pengelolaan data yang lebih rapi dan terstruktur.

2. **Guided 2**

Source Code

```go
package main

import (
    "fmt"
)

// Fungsi untuk mengurutkan array menggunakan Insertion Sort
func insertionSort(arr []int, n int) {
    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 selisih elemen array tetap
func isConstantDifference(arr []int, n int) (bool, int) {
    if n < 2 {
        return true, 0
    }

    difference := arr[1] - arr[0]
    for i := 1; i < n-1; i++ {
        if arr[i+1]-arr[i] != difference {
            return false, 0
        }
    }
    return true, difference
}

func main() {
    var arr []int
    var num int

    // Input data hingga bilangan negatif ditemukan
```

```
    fmt.Println("Masukkan data integer (akhiri dengan bilangan
negatif):")
    for {
        fmt.Scan(&num)
        if num < 0 {
            break
        }
        arr = append(arr, num)
    }

    n := len(arr)

    // Urutkan array menggunakan Insertion Sort
    insertionSort(arr, n)

    // Periksa apakah selisih elemen tetap
    isConstant, difference := isConstantDifference(arr, n)

    // Tampilkan hasil pengurutan
    fmt.Println("Array setelah diurutkan:")
    for _, val := range arr {
        fmt.Printf("%d ", val)
    }
    fmt.Println()

    // Tampilkan status jarak
    if isConstant {
        fmt.Printf("Data berjarak %d\n", difference)
    } else {
        fmt.Println("Data berjarak tidak tetap")
    }
}
```

Screenshoot Output

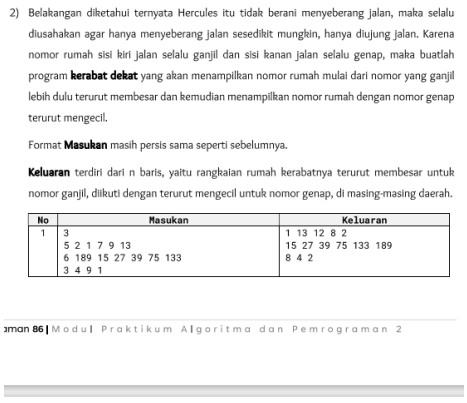
```
PS D:\Kuliah\Matkul\SEMESTER 3\ALPRO PRAKTEK\PRAKTIKUM\MODUL XII>
Masukkan data integer (akhiri dengan bilangan negatif):
31 13 25 43 1 7 19 37 -5
Array setelah diurutkan:
1 7 13 19 25 31 37 43
Data berjarak 6
PS D:\Kuliah\Matkul\SEMESTER 3\ALPRO PRAKTEK\PRAKTIKUM\MODUL XII>
```

Deskripsi Program

Program tersebut memanfaatkan algoritma insertion sort untuk mengurutkan array yang berisi bilangan bulat yang dimasukkan oleh pengguna, dengan proses penginputan dihentikan jika bilangan negatif dimasukkan. Setelah array diurutkan, program memeriksa apakah selisih antar elemen memiliki nilai yang sama atau konstan. Jika selisihnya konsisten, program akan menampilkan nilai selisih tersebut; jika tidak, program akan menyatakan bahwa jarak antar data tidak konsisten. Pengurutan dilakukan dengan cara memindahkan elemen-elemen yang lebih besar dari elemen kunci (key) ke posisi yang lebih tinggi, sehingga menghasilkan array yang terurut secara ascending. Hasil akhir dari program ini mencakup array yang telah diurutkan serta informasi tentang keberadaan pola jarak yang tetap antara elemen-elemen dalam array.

III. Unguided

1. Unguided 1



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.

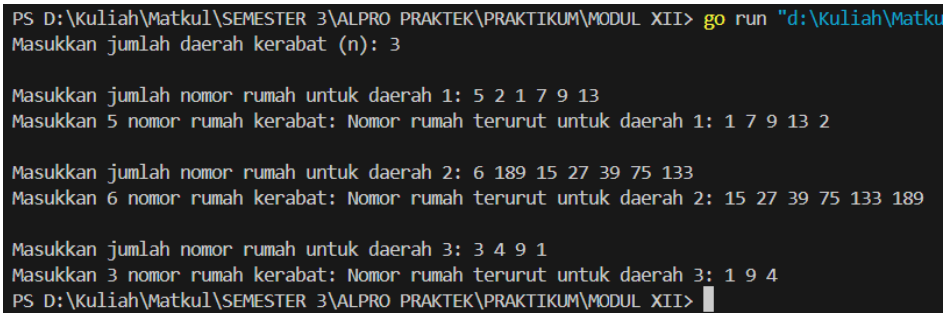
Source Code

```go
package main

import "fmt"

// Fungsi untuk mengurutkan array secara ascending menggunakan
Selection Sort
func selectionSortAscending(arr []int) {
```

```
    n := len(arr)
    for i := 0; i < n-1; i++ {
        minIdx := i
        for j := i + 1; j < n; j++ {
            if arr[j] < arr[minIdx] {
                minIdx = j
            }
        }
        arr[i], arr[minIdx] = arr[minIdx], arr[i]
    }
}

// Fungsi untuk mengurutkan array secara descending menggunakan
Selection Sort
func selectionSortDescending(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] {
                maxIdx = j
            }
        }
        arr[i], arr[maxIdx] = arr[maxIdx], arr[i]
    }
}

// Fungsi untuk memisahkan bilangan ganjil dan genap
func pisahkanGanjilGenap(arr []int) ([]int, []int) {
    var ganjil, genap []int
    for _, num := range arr {
        if num%2 == 0 {
            genap = append(genap, num)
        } else {
            ganjil = append(ganjil, num)
        }
    }
    return ganjil, genap
}

func main() {
    var t_2311102040 int
    fmt.Print("Masukkan jumlah daerah kerabat (n): ")
    fmt.Scan(&t_2311102040 )
```

```
    // Iterasi untuk setiap daerah
    for daerah := 1; daerah <= t_2311102040; daerah++ {
        var m int
        fmt.Printf("\nMasukkan jumlah nomor rumah untuk daerah %d:
", daerah)
        fmt.Scan(&m)

        arr := make([]int, m)
        fmt.Printf("Masukkan %d nomor rumah kerabat: ", m)
        for i := 0; i < m; i++ {
            fmt.Scan(&arr[i])
        }

        // Pisahkan nomor ganjil dan genap
        ganjil, genap := pisahkanGanjilGenap(arr)

        // Urutkan ganjil secara naik dan genap secara turun
        selectionSortAscending(ganjil)
        selectionSortDescending(genap)

        // Tampilkan hasil
        fmt.Printf("Nomor rumah terurut untuk daerah %d: ", daerah)
        for _, num := range ganjil {
            fmt.Printf("%d ", num)
        }
        for _, num := range genap {
            fmt.Printf("%d ", num)
        }
        fmt.Println()
    }
}
```

Screenshot Output



Deskripsi Program

Program ini dibuat untuk mengurutkan nomor rumah kerabat dengan aturan tertentu. Nomor rumah ganjil diurutkan dari yang terkecil ke yang terbesar (ascending), sedangkan nomor rumah genap diurutkan dari yang terbesar ke yang terkecil (descending). Pengguna diminta memasukkan jumlah daerah kerabat, lalu untuk setiap daerah, memasukkan jumlah nomor rumah dan daftar nomor rumah tersebut. Setelah itu, program memisahkan nomor ganjil dan genap, lalu mengurutkan masing-masing menggunakan cara selection sort. Hasil akhirnya, program menampilkan nomor rumah ganjil yang sudah terurut diikuti nomor rumah genap yang juga sudah diurutkan, untuk setiap daerah.

2. Unguided 2

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"
    "strconv"
    "strings"
)

func main() {
    scanner := bufio.NewScanner(os.Stdin)
    fmt.Println("Masukkan angka-angka (akhiri dengan -5313):")

    var data_2311102040 []int

    // Membaca input dari pengguna
    for scanner.Scan() {
        line := scanner.Text()
        numStrings := strings.Fields(line)
        for _, numStr := range numStrings {
            num, err := strconv.Atoi(numStr)
            if err != nil {
                fmt.Println("Input tidak valid!")
                return
            }
            if num == -5313 {
                // Mengakhiri program jika ditemukan -5313
                return
            }
            if num == 0 {
                // Jika input adalah 0, hitung median dari data yang
sudah terkumpul
                if len(data_2311102040) == 0 {
                    fmt.Println("Data kosong!")
                } else {
                    selectionSort(data_2311102040)
                    median := calculateMedian(data_2311102040)
                    fmt.Println(median)
                }
            } else {
                // Tambahkan bilangan ke dalam data
                data_2311102040 = append(data_2311102040, num)
```

```
            }
        }
    }
}

// Fungsi untuk menghitung median
func calculateMedian(data_2311102040 []int) int {
    n := len(data_2311102040)
    if n%2 == 1 {
        // Jika jumlah data ganjil, median adalah nilai tengah
        return data_2311102040[n/2]
    }
    // Jika jumlah data genap, median adalah rata-rata dari dua
nilai tengah
    return (data_2311102040[n/2-1] + data_2311102040[n/2]) / 2
}

// Fungsi untuk mengurutkan data menggunakan selection sort
func selectionSort(data_2311102040 []int) {
    n := len(data_2311102040)
    for i := 0; i < n-1; i++ {
        // Temukan indeks elemen terkecil di bagian yang belum
diurutkan
        minIdx := i
        for j := i + 1; j < n; j++ {
            if data_2311102040[j] < data_2311102040[minIdx] {
                minIdx = j
            }
        }
        // Tukar elemen terkecil dengan elemen di posisi i
        data_2311102040[i], data_2311102040[minIdx] =
data_2311102040[minIdx], data_2311102040[i]
    }
}
```

Screenshot Output

```
PS D:\Kuliah\Matkul\SEMESTER 3\ALPRO PRAKTEK\PRAKTIKUM\MODUL XII> go
Masukkan angka-angka (akhiri dengan -5313):
7 23 11 0 5 19 2 29 3 13 17 0 -5313
11
12
PS D:\Kuliah\Matkul\SEMESTER 3\ALPRO PRAKTEK\PRAKTIKUM\MODUL XII>
```

Deskripsi Program

Kode program di atas adalah solusi untuk menghitung nilai median dari data masukan berdasarkan aturan yang diberikan. Program membaca data secara bertahap dari input pengguna, dengan setiap angka diproses satu per satu. Angka 0 digunakan untuk memicu perhitungan median dari data yang sudah terkumpul hingga titik tersebut, sementara angka -5313 digunakan sebagai tanda untuk mengakhiri program. Program menggunakan algoritma selection sort untuk mengurutkan data sebelum menghitung median, yang dilakukan dengan mencari elemen terkecil secara iteratif dan menukarnya ke posisi yang benar. Setelah data terurut, median dihitung berdasarkan jumlah data: jika jumlahnya ganjil, nilai tengah diambil, dan jika genap, rata-rata dua nilai tengah dihitung (dibulatkan ke bawah).

3. Unguided 3

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.}
```

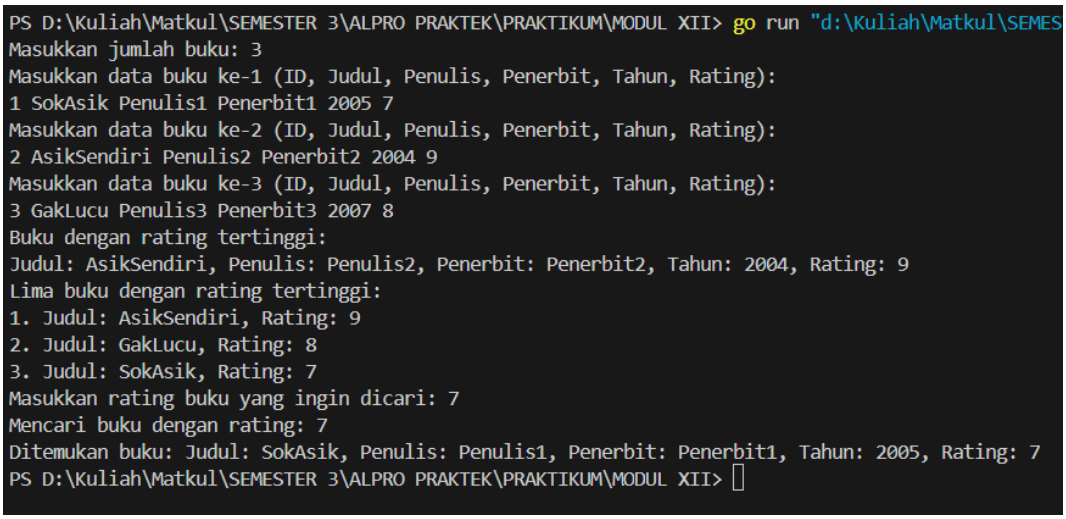
Source Code

```
package main

import (
    "fmt"
)

// Definisi struct Buku
type Buku struct {
    ID       int
    Judul    string
    Penulis  string
    Penerbit string
    Tahun    int
    Rating   int
}

// Definisi tipe array untuk menyimpan data buku
const nMax = 7919

type DaftarBuku [nMax]Buku

// Subprogram untuk memasukkan data buku
func DaftarkanBuku(pustaka *DaftarBuku, n *int) {
    fmt.Print("Masukkan jumlah buku: ")
    fmt.Scan(n)
    for i := 0; i < *n; i++ {
        fmt.Printf("Masukkan data buku ke-%d (ID, Judul, Penulis,
Penerbit, Tahun, Rating):\n", i+1)
        fmt.Scan(&pustaka[i].ID, &pustaka[i].Judul,
&pustaka[i].Penulis, &pustaka[i].Penerbit, &pustaka[i].Tahun,
&pustaka[i].Rating)
    }
}

// Subprogram untuk mencetak buku dengan rating tertinggi
func CetakTerfavorit(pustaka DaftarBuku, n int) {
    if n == 0 {
        fmt.Println("Tidak ada buku yang terdaftar.")
        return
    }

    // Cari buku dengan rating tertinggi
    tertinggi := pustaka[0].Rating
```

```
    var index int
    for i := 1; i < n; i++ {
        if pustaka[i].Rating > tertinggi {
            tertinggi = pustaka[i].Rating
            index = i
        }
    }

    fmt.Println("Buku dengan rating tertinggi:")
    fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s, Tahun: %d,
Rating: %d\n",
        pustaka[index].Judul, pustaka[index].Penulis,
pustaka[index].Penerbit, pustaka[index].Tahun,
pustaka[index].Rating)
}

// Subprogram untuk mengurutkan data buku berdasarkan rating
(Insertion Sort)
func UrutBuku(pustaka *DaftarBuku, n int) {
    for i := 1; i < n; i++ {
        key := pustaka[i]
        j := i - 1

        // Geser elemen yang lebih kecil dari key ke kanan
        for j >= 0 && pustaka[j].Rating < key.Rating {
            pustaka[j+1] = pustaka[j]
            j--
        }
        pustaka[j+1] = key
    }
}

// Subprogram untuk mencetak lima buku dengan rating tertinggi
func Cetak5Terbaik(pustaka DaftarBuku, n int) {
    fmt.Println("Lima buku dengan rating tertinggi:")
    for i := 0; i < 5 && i < n; i++ {
        fmt.Printf("%d. Judul: %s, Rating: %d\n", i+1,
pustaka[i].Judul, pustaka[i].Rating)
    }
}

// Subprogram untuk mencari buku berdasarkan rating tertentu
func CariBuku(pustaka DaftarBuku, n int, ratingCari int) {
    fmt.Printf("Mencari buku dengan rating: %d\n", ratingCari)
    found := false
```

```
    for i := 0; i < n; i++ {
        if pustaka[i].Rating == ratingCari {
            fmt.Printf("Ditemukan buku: Judul: %s, Penulis: %s,
Penerbit: %s, Tahun: %d, Rating: %d\n",
                pustaka[i].Judul, pustaka[i].Penulis,
pustaka[i].Penerbit, pustaka[i].Tahun, pustaka[i].Rating)
            found = true
        }
    }
    if !found {
        fmt.Println("Tidak ada buku dengan rating tersebut.")
    }
}

func main() {
    var pustaka DaftarBuku
    var n_2311102040 int

    // Memasukkan data buku
    DaftarkanBuku(&pustaka, &n_2311102040)

    // Mengurutkan buku berdasarkan rating
    UrutBuku(&pustaka, n_2311102040)

    // Menampilkan buku dengan rating tertinggi
    CetakTerfavorit(pustaka, n_2311102040)

    // Menampilkan lima buku dengan rating tertinggi
    Cetak5Terbaik(pustaka, n_2311102040)

    // Mencari buku dengan rating tertentu
    var ratingCari int
    fmt.Print("Masukkan rating buku yang ingin dicari: ")
    fmt.Scan(&ratingCari)
    CariBuku(pustaka, n_2311102040, ratingCari)
}
```

Screenshot Output



Deskripsi Program

Program ini adalah aplikasi sederhana untuk mengelola data buku di perpustakaan. Data setiap buku, seperti ID, judul, penulis, penerbit, tahun, dan rating, disimpan dalam struktur data bernama Buku. Program ini menyediakan beberapa fungsi, seperti memasukkan data buku, mengurutkan buku berdasarkan rating menggunakan metode insertion sort, dan menampilkan buku dengan rating tertinggi. Selain itu, program juga dapat mencetak lima buku dengan rating tertinggi serta mencari buku berdasarkan rating tertentu yang dimasukkan oleh pengguna. Dengan cara ini, program membantu pengguna untuk mengelola dan