LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 12

PENGURUTAN DATA

Disusun Oleh : Brian Farrel Evandhika

2311102037

IF 11 06

Dosen Pengampu :

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

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

12.1 Ide Algorltma Selection Sort

Pengurutan secara seleksi ini idenya adalah mencari nilai ekstrim pada sekumpulan data, kemudian

meletakkan pada posisi yang seharusnya. Pada penjelasan berikut ini data akan diurut membesar

(ascending), dan data dengan indeks kecil ada di "kiri" dan indeks besar ada di "kanan".

1) Cari nilai terkecil di dalam rentang data tersisa

2) Pindahkanltukar tempat dengan data yang berada pada posisi paling kiri pada rentang data

tersisa tersebut.

3) utangi proses ini sampai tersisa hanya satu data saja.

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.

```
i = 1
for i <= n-1 {
   idx_min = i - 1
   for j = i; j < n; j++ {
      if a[idx_min] > a[j] {
         idx_min = j
      }
   }
   t = a[idx_min]
   a[idx_min] = a[i-1]
   a[i-1] = t
   i = i + 1
```

```
}
```

12.2 Algoritma Selection Sort

Adapun algoritma selection sort pada untuk mengurutkan array bertipe data bilangan bulat secara

membesar atau ascending adalah sebagai berikut ini!

```
type arrInt [4321]int


func selectionSort(T *arrInt, n int) {
	/* I.S. terdefinisi array T yang berisi n bilangan bulat
	 * F.S. array T terurut secara ascending atau membesar dengan SELECTION SORT
*/
	var t, i, j, idx_min int
	for i = 1; i <= n-1; i++ {
		idx_min = i - 1
		for j = i; j < n; j++ {
			if T[idx_min] > T[j] {
				idx_min = j
			}
		}
		t = T[idx_min]
		T[idx_min] = T[i-1]
		T[i-1] = t
	}
}
```

Sama halnya apabila array yang akan diurutkan adalah bertipe data struct, maka tambahkan field pada

saat proses perbandingan nilai ekstrim, kemudian tipe data dari variabel t sama dengan struct dari

arraynya.

```
type mahasiswa struct {
	nama, nim, kelas, jurusan string
	ipk float64
}
type arrMhs [2023]mahasiswa
func selectionSort2(T *arrMhs, n int) {
	/* I.S. terdefinisi array T yang berisi n data mahasiswa
	 * F.S. array T terurut secara ascending atau membesar berdasarkan ipk dengan
	 * menggunakan algoritma SELECTION SORT */

	var i, j, idx_min int
	var t mahasiswa

	for i = 1; i <= n-1; i++ {
		idx_min = i - 1
		for j = i; j < n; j++ {
			if T[idx_min].ipk > T[j].ipk {
				idx_min = j
			}
		}
		t = T[idx_min]
		T[idx_min] = T[i-1]
		T[i-1] = t
	}
}
```

GUIDED

Soal Studi Case 1

Sourcecode

```
//2311102037_BRIAN FARREL EVANDHIKA_IF 11_06
package main

import (
    "fmt"
)

// Fungsi untuk mengurutkan array menggunakan Selection Sort
func selectionSort(arr []int, n int) {
    for i := 0; i < n-1; i++ {
        idxMin := i
        for j := i + 1; j < n; j++ {
            // Cari elemen terkecil
            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 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)

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

Screenshot Program

```
PS C:\Users\MSI GAMING> go run "c:\Users\MSI GAMING\Document
s\TELKOM UNIVERSITY\SEMESTER 3\PRAKTIKUM ALGORITMA DAN PEMRO
GRAMAN 2\Pertemuan Ke 12\Guided-1.go"
Masukkan jumlah daerah kerabat (n): 2

Masukkan jumlah nomor rumah kerabat untuk daerah 1: 2
Masukkan 2 nomor rumah kerabat: 1
2
Nomor rumah terurut untuk daerah 1: 1 2

Masukkan jumlah nomor rumah kerabat untuk daerah 2: 3
Masukkan 3 nomor rumah kerabat: 3
4
5
Nomor rumah terurut untuk daerah 2: 3 4 5
PS C:\Users\MSI GAMING>
```

Deskripsi Program

Program di atas mengimplementasikan algoritma Selection Sort untuk mengurutkan nomor rumah kerabat dalam berbagai daerah. Pertama, program meminta pengguna untuk memasukkan jumlah daerah yang ingin diolah. Kemudian, untuk setiap daerah, pengguna diminta untuk memasukkan jumlah nomor rumah yang akan diurutkan. Setelah itu, pengguna menginput nomor rumah tersebut, dan program mengurutkannya dari terkecil ke terbesar menggunakan algoritma Selection Sort. Hasil pengurutan ditampilkan kembali kepada pengguna untuk setiap daerah. Program ini menggambarkan cara dasar mengurutkan angka dengan Selection Sort dalam bahasa pemrograman Go.

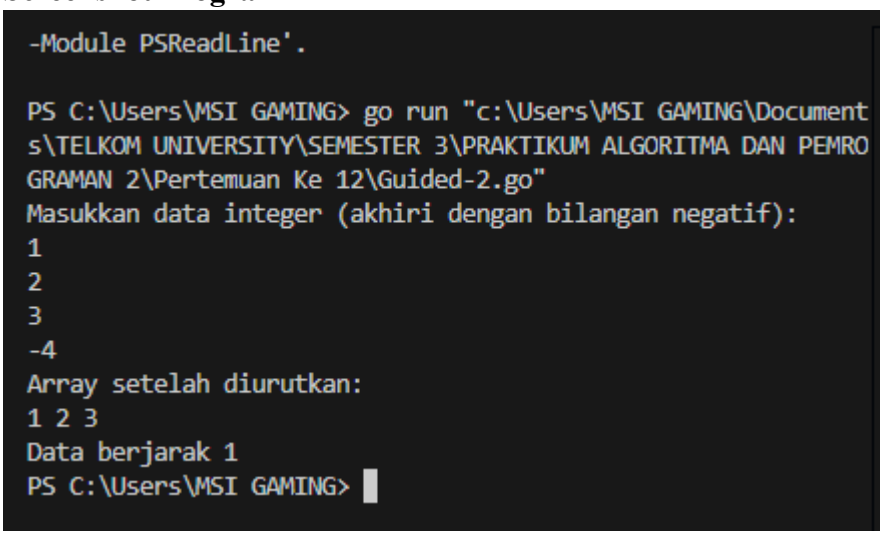
Soal Studi Case 2

Source Code

```go
//2311102037_BRIAN FARREL EVANDHIKA_IF 11_06
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")
    }
}
```

Screenshot Program



Deskripsi Program

Program di atas mengimplementasikan algoritma Insertion Sort untuk mengurutkan array integer yang dimasukkan oleh pengguna sampai bilangan negatif ditemukan. Setelah array diurutkan, program juga memeriksa apakah selisih antara elemen-elemen yang berurutan dalam array tersebut tetap konstan. Jika selisih elemen-elemen tersebut tetap, program menampilkan selisih tersebut; jika tidak, program menampilkan bahwa jarak elemen tidak tetap. Hasil pengurutan dan status jarak elemen ditampilkan kepada pengguna setelah proses selesai.

II. UNGUIDED

Soal Studi Case 1

Buatlah sebuah program yang digunakan untuk membaca data integer seperti contoh yang
diberikan di bawah ini, kemudian diurutkan (menggunakan metoda insertion sort), dan
memeriksa apakah data yang terurut berjarak sama terhadap data sebelumnya.
Masukan terdiri dari sekumpulan bilangan bulat yang diakhiri Oleh bilangan negatif. Hanya
bilangan non negatif saja yang disimpan ke dalam array.
Keluaran terdiri dari dua baris. Baris pertama adalah isi dari array setelah dilakukan
pengurutan, sedangkan baris kedua adalah status Jarak setiap bilangan yang ada di dalam
array. "Data berjarak x" atau "data berjarak tidak tetap".

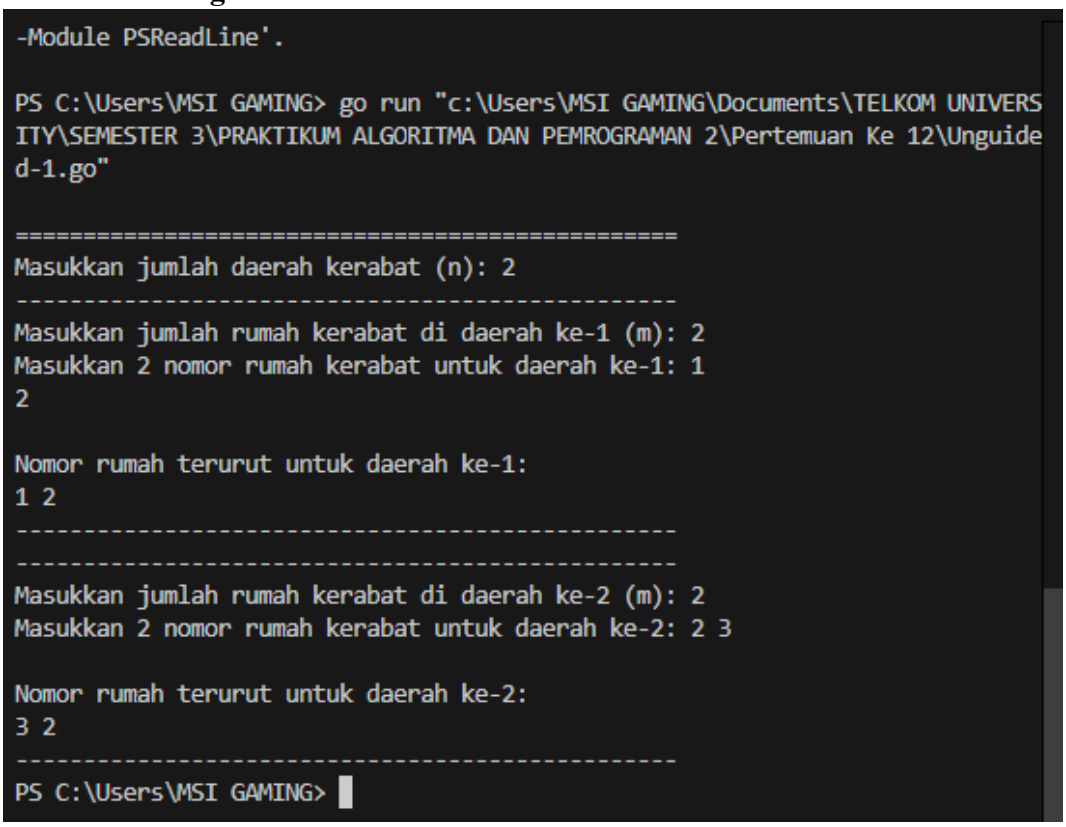
Source Code

```
//2311102037_BRIAN FARREL EVANDHIKA_IF 11_06
package main

import "fmt"

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

func printSeparator() {
    fmt.Println("\n==================================================
")
}

func printDashedSeparator() {
```

```
    fmt.Println("------------------------------------------------")
}

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

    var odd, even []int
    for _, num := range arr {
        if num%2 == 0 {
            even = append(even, num)
        } else {
            odd = append(odd, num)
        }
    }
    return odd, even
}

func printSortedResults(i int, odd, even []int) {

    selectionSort(odd, true)
    selectionSort(even, false)

    fmt.Printf("\nNomor rumah terurut untuk daerah ke-%d:\n", i+1)

    for _, num := range odd {
        fmt.Printf("%d ", num)
    }

    for _, num := range even {
        fmt.Printf("%d ", num)
    }
    fmt.Println()
}

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

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

        odd, even := processDaerah(i, m)

        printSortedResults(i, odd, even)
        printDashedSeparator()
    }
}
```

Screenshot Program



Deskripsi Program

rogram ini mengimplementasikan algoritma Selection Sort untuk mengurutkan nomor rumah kerabat dalam berbagai daerah dengan cara yang unik. Pengguna diminta untuk memasukkan jumlah daerah, dan untuk setiap daerah, pengguna harus memasukkan jumlah nomor rumah yang akan diurutkan. Setiap nomor rumah kemudian dikategorikan menjadi bilangan ganjil dan genap. Bilangan ganjil diurutkan secara menaik, sedangkan

bilangan genap diurutkan secara menurun menggunakan algoritma Selection Sort. Hasil pengurutan ditampilkan kembali kepada pengguna dengan pemisahan yang jelas menggunakan pemisah berbentuk garis. Program ini memperlihatkan penggunaan algoritma pengurutan dan manipulasi data dalam bahasa pemrograman Go dengan antarmuka pengguna yang interaktif.

Soal Studi Case 2

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 selurufi data yang sudah
terbaca,jika data yang dibaca saat itu adalah O.
Masukan berbentuk rangkaian bilangan bulat. Masukan tidak akan berisi lebih dari 1000000
data, tidak termasuk bilangan O. Data O 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.

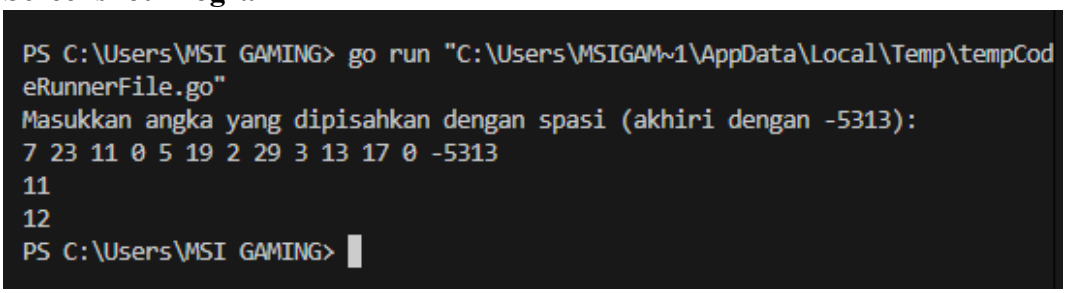
Source Code

```go
// 2311102037_BRIAN FARREL EVANDHIKA_IF 11_06
package main

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

```
)

// Fungsi untuk mengurutkan array menggunakan algoritma Selection
Sort
func sortArray(arr []int) {
    for i := 0; i < len(arr)-1; i++ {
        minIndex := i
        for j := i + 1; j < len(arr); j++ {
            if arr[j] < arr[minIndex] {
                minIndex = j
            }
        }
        arr[i], arr[minIndex] = arr[minIndex], arr[i]
    }
}

// Fungsi untuk menghitung median dari array yang telah diurutkan
func getMedian(arr []int) float64 {
    length := len(arr)
    if length%2 == 0 {
        return float64(arr[length/2-1]+arr[length/2]) / 2.0
    }
    return float64(arr[length/2])
}

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

    reader.Scan()
    input := reader.Text()
    values := strings.Split(input, " ")
    var numbers []int

    for _, val := range values {
        number, err := strconv.Atoi(val)
        if err != nil {
            fmt.Println("Input tidak valid, masukkan hanya angka
bulat.")
            return
        }

        if number == -5313 {
            break
```

```
        } else if number == 0 {
            sortArray(numbers)
            median := getMedian(numbers)
            fmt.Printf("%.0f\n", median)
        } else {
            numbers = append(numbers, number)
        }
    }
}
```

Screenshot Program



Deskripsi Program

Program ini mengimplementasikan algoritma Selection Sort untuk mengurutkan array integer yang dimasukkan oleh pengguna hingga menemukan nilai penanda -5313. Pengguna diminta untuk memasukkan angka yang dipisahkan dengan spasi, dan jika angka 0 dimasukkan, program akan mengurutkan array yang telah dimasukkan hingga saat itu dan menghitung median dari array yang terurut. Jika input bukan angka atau nilai negatif lain selain -5313 dimasukkan, program akan berhenti. Hasil median ditampilkan dalam format bulat tanpa desimal. Program ini menggabungkan pengurutan dengan Selection Sort dan perhitungan median dalam bahasa pemrograman Go dengan antarmuka pengguna yang interaktif dan tanggapan berbasis teks.

Soal Studi Case 3

Sebuah program perpustakaan digunakan untuk mengelola data bukWdi dalanv 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.

Source Code

```
// 2311102037_BRIAN FARREL EVANDHIKA_IF 11_06
package main

import (
    "fmt"
)

type Buku struct {
    id        int
    judul     string
    penulis   string
    penerbit  string
    eksemplar int
    tahun     int
    rating    int
}

func DaftarkanBuku(perpustakaan *[]Buku, jumlah int) {
    for i := 0; i < jumlah; i++ {
        var buku Buku
        fmt.Printf("Masukkan data untuk buku ke-%d (id, judul,
penulis, penerbit, eksemplar, tahun, rating):\n", i+1)
        fmt.Scan(&buku.id, &buku.judul, &buku.penulis,
&buku.penerbit, &buku.eksemplar, &buku.tahun, &buku.rating)
        *perpustakaan = append(*perpustakaan, buku)
    }
}

func CetakBukuRatingTertinggi(perpustakaan []Buku) {
    if len(perpustakaan) == 0 {
        fmt.Println("Perpustakaan kosong.")
```

```
        return
    }
    bukuFavorit := perpustakaan[0]
    for _, buku := range perpustakaan {
        if buku.rating > bukuFavorit.rating {
            bukuFavorit = buku
        }
    }
    fmt.Println("Buku dengan rating tertinggi:")
    fmt.Printf("ID: %d, Judul: %s, Penulis: %s, Penerbit: %s,
Eksemplar: %d, Tahun: %d, Rating: %d\n",
        bukuFavorit.id, bukuFavorit.judul, bukuFavorit.penulis,
bukuFavorit.penerbit, bukuFavorit.eksemplar, bukuFavorit.tahun,
bukuFavorit.rating)
}

func UrutkanBukuBerdasarkanRating(perpustakaan *[]Buku) {
    for i := 1; i < len(*perpustakaan); i++ {
        kunci := (*perpustakaan)[i]
        j := i - 1
        for j >= 0 && (*perpustakaan)[j].rating < kunci.rating {
            (*perpustakaan)[j+1] = (*perpustakaan)[j]
            j--
        }
        (*perpustakaan)[j+1] = kunci
    }
}

func Cetak5BukuTerbaik(perpustakaan []Buku) {
    fmt.Println("5 Buku dengan rating tertinggi:")
    for i := 0; i < 5 && i < len(perpustakaan); i++ {
        buku := perpustakaan[i]
        fmt.Printf("ID: %d, Judul: %s, Penulis: %s, Penerbit: %s,
Eksemplar: %d, Tahun: %d, Rating: %d\n",
            buku.id, buku.judul, buku.penulis, buku.penerbit,
buku.eksemplar, buku.tahun, buku.rating)
    }
}

func CariBukuBerdasarkanRating(perpustakaan []Buku, targetRating
int) {
    ditemukan := false
    for _, buku := range perpustakaan {
        if buku.rating == targetRating {
            ditemukan = true
```

```
            fmt.Printf("ID: %d, Judul: %s, Penulis: %s, Penerbit:
%s, Eksemplar: %d, Tahun: %d, Rating: %d\n",
                buku.id, buku.judul, buku.penulis, buku.penerbit,
buku.eksemplar, buku.tahun, buku.rating)
        }
    }
    if !ditemukan {
        fmt.Println("Tidak ada buku dengan rating tersebut.")
    }
}

func main() {
    var jumlahBuku int
    fmt.Print("Masukkan jumlah buku di perpustakaan: ")
    fmt.Scan(&jumlahBuku)

    if jumlahBuku <= 0 || jumlahBuku > 7919 {
        fmt.Println("Jumlah buku harus antara 1 hingga 7919.")
        return
    }

    var perpustakaan []Buku

    DaftarkanBuku(&perpustakaan, jumlahBuku)
    CetakBukuRatingTertinggi(perpustakaan)
    UrutkanBukuBerdasarkanRating(&perpustakaan)
    Cetak5BukuTerbaik(perpustakaan)

    var ratingDicari int
    fmt.Print("Masukkan rating buku yang ingin dicari: ")
    fmt.Scan(&ratingDicari)
    CariBukuBerdasarkanRating(perpustakaan, ratingDicari)
}
```

Screenshot Program

```
PS C:\Users\MSI GAMING> go run "C:\Users\MSIGAM~1\AppData\Local\Temp\tempCodeRunnerFile.go"
Masukkan jumlah buku di perpustakaan: 1
Masukkan data untuk buku ke-1 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
1 Bagus Agus Sugus 100 1979 1
Buku dengan rating tertinggi:
ID: 1, Judul: Bagus, Penulis: Agus, Penerbit: Sugus, Eksemplar: 100, Tahun: 1979, Rating: 1
Lima buku dengan rating tertinggi:
ID: 1, Judul: Bagus, Penulis: Agus, Penerbit: Sugus, Eksemplar: 100, Tahun: 1979, Rating: 1
Masukkan rating buku yang ingin dicari: 1
ID: 1, Judul: Bagus, Penulis: Agus, Penerbit: Sugus, Eksemplar: 100, Tahun: 1979, Rating: 1
PS C:\Users\MSI GAMING>
```

Deskripsi Program

Program ini mengelola data buku dalam perpustakaan, memungkinkan pengguna untuk memasukkan sejumlah buku dengan berbagai atribut seperti id, judul, penulis, penerbit, eksemplar, tahun, dan rating. Program menyediakan fungsi untuk mendaftarkan buku, mencetak buku dengan rating tertinggi, mengurutkan buku berdasarkan rating secara menurun, mencetak lima buku terbaik, dan mencari buku berdasarkan rating tertentu. Pengguna dapat menginput jumlah buku yang akan dimasukkan dan program akan memproses data tersebut, menampilkan buku dengan rating tertinggi dan lima buku dengan rating terbaik, serta mencari buku yang sesuai dengan rating yang dicari. Program ini ditulis dalam bahasa pemrograman Go dan menyediakan antarmuka pengguna berbasis teks yang interaktif.