LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL XII dan XIII

PENGURUTAN DATA

Disusun Oleh :

PETRA PRIADI S.P GINTING (2311102273)

IF-06

Dosen Pengampu :

ABEDNEGO DWI SEPTIADI

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Pengurutan data adalah operasi penting dalam pengolahan koleksi data seperti array atau slice. Bahasa Go menyediakan paket bawaan bernama sort untuk mempermudah pengurutan berbagai jenis data.

1. Paket sort

Paket sort dalam Go adalah alat utama untuk pengurutan data. Dengan fungsi-fungsi bawaan di dalamnya, Anda dapat mengurutkan slice angka, string, atau bahkan data dengan aturan khusus.

2. Pengurutan Standar

Paket sort menyediakan fungsi khusus untuk mengurutkan data slice bertipe bawaan:

- **sort.Ints**: Untuk mengurutkan slice bertipe integer secara ascending.
- **sort.Strings**: Untuk mengurutkan slice string secara alfabetis.
- **sort.Float64s**: Untuk mengurutkan slice float secara ascending.

3. Pengurutan Custom

Untuk kebutuhan yang lebih kompleks, seperti pengurutan berdasarkan aturan tertentu (contohnya descending atau pengurutan berdasarkan atribut pada slice struct), Go menyediakan fungsi sort.Slice. Fungsi ini memungkinkan Anda menentukan logika pengurutan menggunakan fungsi comparator.

4. Pengurutan Slice Struct

Dalam kasus data kompleks seperti slice berisi struct, sort.Slice digunakan untuk menentukan atribut mana yang menjadi kriteria pengurutan. Anda dapat membuat fungsi pembanding untuk menentukan bagaimana elemen-elemen di dalam slice dibandingkan satu sama lain.

5. Karakteristik Pengurutan di Go

- **In-place sorting**: Pengurutan di Go dilakukan secara in-place, artinya slice asli akan dimodifikasi tanpa membuat salinan baru.
- **Ascending sebagai default**: Pengurutan bawaan berjalan secara menaik (ascending), kecuali Anda membuat logika khusus untuk pengurutan menurun (descending).

6. Keunggulan Paket sort

- Mudah digunakan dengan fungsi bawaan.
- Fleksibel untuk kebutuhan pengurutan kompleks.
- Performa tinggi, karena didesain untuk menangani data dalam skala besar.

Kesimpulan

Paket sort di Go adalah alat yang sangat kuat dan fleksibel untuk mengatur data. Dengan fungsi bawaan yang sederhana dan kemampuan untuk mengatur pengurutan custom, pengurutan data menjadi operasi yang efisien dan mudah dikelola.

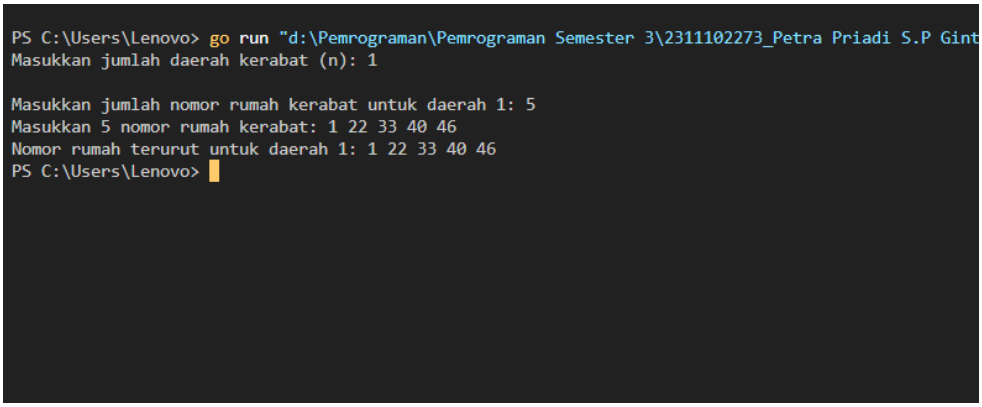
II. GUIDED

1. **Sourcecode**

```
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()
      }
}
```

Screenshoot Output



Deskripsi Program

Pengguna diminta untuk memasukkan jumlah daerah (n) yang akan diproses. Untuk setiap daerah, pengguna memberikan jumlah nomor rumah (m) dan nomor-nomor rumah tersebut. Nomor rumah yang dimasukkan oleh pengguna disimpan dalam sebuah array, lalu diurutkan menggunakan algoritma Selection Sort, di mana elemen terkecil dipindahkan ke posisi awal secara bertahap hingga seluruh array terurut. Setelah proses pengurutan selesai, program menampilkan daftar nomor rumah yang sudah terurut untuk setiap daerah.

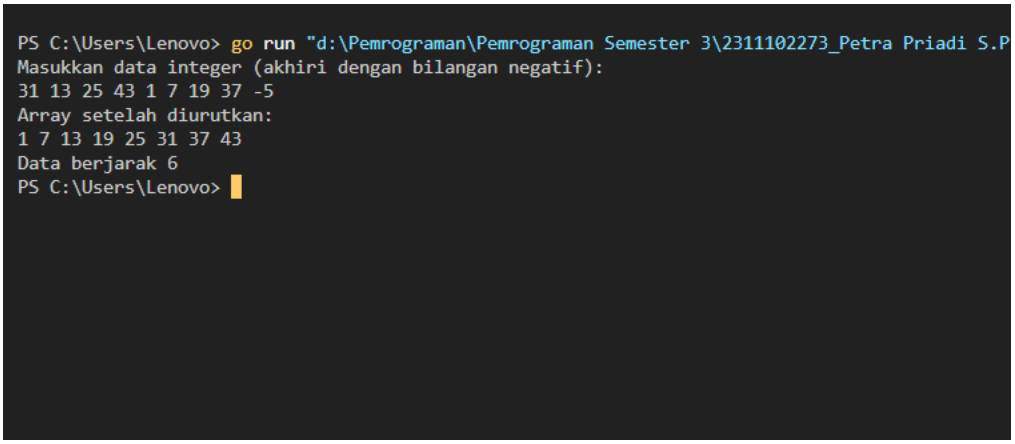
2. **Sourcecode**

```
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



Deskripsi Program

Program meminta pengguna untuk memasukkan sejumlah bilangan bulat satu per satu, dengan proses berakhir ketika bilangan negatif dimasukkan. Bilangan-bilangan ini disimpan dalam array dan diurutkan secara menaik menggunakan algoritma Insertion Sort, di mana elemen-elemen array disisipkan di posisi yang sesuai melalui penggeseran elemen yang lebih besar. Setelah array diurutkan, program memeriksa apakah selisih antara elemen yang berurutan tetap konstan menggunakan fungsi isConstantDifference. Jika konstan, program menampilkan selisihnya; jika tidak, ia menyatakan bahwa data tidak memiliki selisih yang tetap.

III. UNGUIDED

1. Soal Studi Case

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.

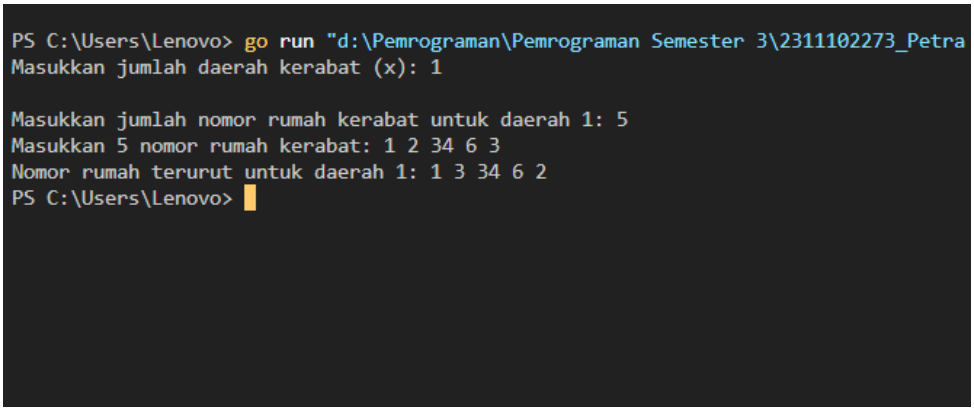
No	Masukan	Keluaran
1	3 5 2 1 7 9 13 6 189 15 27 39 75 133 3 4 9 1	1 13 12 8 2 15 27 39 75 133 189 8 4 2

Sourcecode

```
package main

import (
	"fmt"
)

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

func main() {
	var x int
	fmt.Print("Masukkan jumlah daerah kerabat (x): ")
	if _, err := fmt.Scan(&x); err != nil || x <= 0 {
```

```
            fmt.Println("Input jumlah daerah harus berupa
angka lebih dari 0.")
            return
      }

      for daerah := 1; daerah <= x; daerah++ {
            var m int
            fmt.Printf("\nMasukkan jumlah nomor rumah
kerabat untuk daerah %d: ", daerah)
            if _, err := fmt.Scan(&m); err != nil || m <=
0 {
                  fmt.Println("Input jumlah nomor rumah
harus berupa angka lebih dari 0.")
                  return
            }
            arr := make([]int, m)
            ganjil := []int{}
            genap := []int{}
            fmt.Printf("Masukkan %d nomor rumah kerabat:
", m)
            for i := 0; i < m; i++ {
                  if _, err := fmt.Scan(&arr[i]); err !=
nil {
                        fmt.Println("Input nomor rumah
tidak valid. Harus berupa angka.")
                        return
                  }
                  if arr[i]%2 == 0 {
                        genap = append(genap, arr[i])
                  } else {
                        ganjil = append(ganjil, arr[i])
                  }
            }

            pengurutanNomorRumah(ganjil, true)
            pengurutanNomorRumah(genap, false)

            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()
      }
}
```

Screenshoot Output



Deskripsi Program

Program meminta pengguna untuk memasukkan jumlah daerah (x), dan untuk setiap daerah, memasukkan jumlah nomor rumah (m) beserta daftar nomor rumahnya. Nomor rumah tersebut dipisahkan menjadi dua kelompok: ganjil dan genap. Kelompok ganjil diurutkan secara menaik (ascending), sedangkan kelompok genap diurutkan secara menurun (descending), menggunakan fungsi pengurutanNomorRumah, yang mengimplementasikan algoritma Selection Sort dengan fleksibilitas untuk menentukan arah pengurutan. Setelah pengelompokan dan pengurutan selesai, program mencetak nomor rumah yang sudah terurut, dengan nomor ganjil terlebih dahulu diikuti oleh nomor genap, untuk setiap daerah.

2. Studi Case

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.

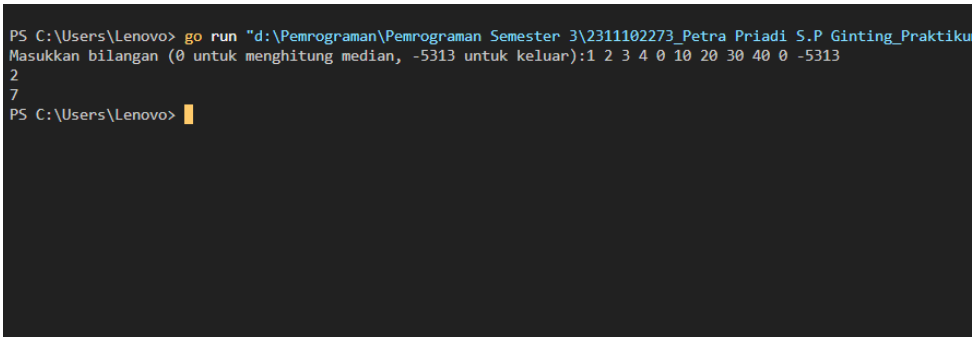
Sourcecode

```
package main


import (
	"fmt"
)


func main() {
	var angka []int
	var input int
```

```
	fmt.Print("Masukkan bilangan (0 untuk menghitung
median, -5313 untuk keluar):")


	for {
		_, err := fmt.Scan(&input)
		if err != nil {
			fmt.Println("Error membaca input:", err)
			break
		}

		if input == -5313 {
			break
		}

		if input == 0 {
			if len(angka) > 0 {
				median := hitungMedian(angka)
				fmt.Println(median)
			}
			continue
		}

		angka = append(angka, input)
	}
}

func hitungMedian(nums []int) int {
```

```
        for i := 1; i < len(nums); i++ {
                key := nums[i]
                j := i - 1

                for j >= 0 && nums[j] > key {
                        nums[j+1] = nums[j]
                        j--
                }
                nums[j+1] = key
        }
        n := len(nums)
        if n%2 == 1 {
                return nums[n/2]
        } else {
                return (nums[n/2-1] + nums[n/2]) / 2
        }
}
```

Screenshoot Output



Deskripsi Program

Pengguna memasukkan sekumpulan bilangan bulat secara interaktif, menghitung median dari bilangan yang sudah dimasukkan ketika angka 0 dimasukkan, dan keluar dari program ketika angka -5313 dimasukkan. Median dihitung menggunakan fungsi hitungMedian, yang pertama-tama mengurutkan array menggunakan Insertion Sort. Jika jumlah elemen ganjil, median adalah elemen di tengah; jika genap, median adalah rata-rata dari dua elemen tengah. Program membaca bilangan satu per satu melalui input pengguna, menyimpannya dalam slice angka, dan memproses input berdasarkan nilai yang dimasukkan: angka 0 memicu penghitungan median, sedangkan angka -5313 menghentikan program. Kode ini mencakup validasi untuk membaca input pengguna dan memastikan penghitungan median hanya dilakukan ketika terdapat setidaknya satu bilangan dalam slice.

3. **Studi Case**

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.

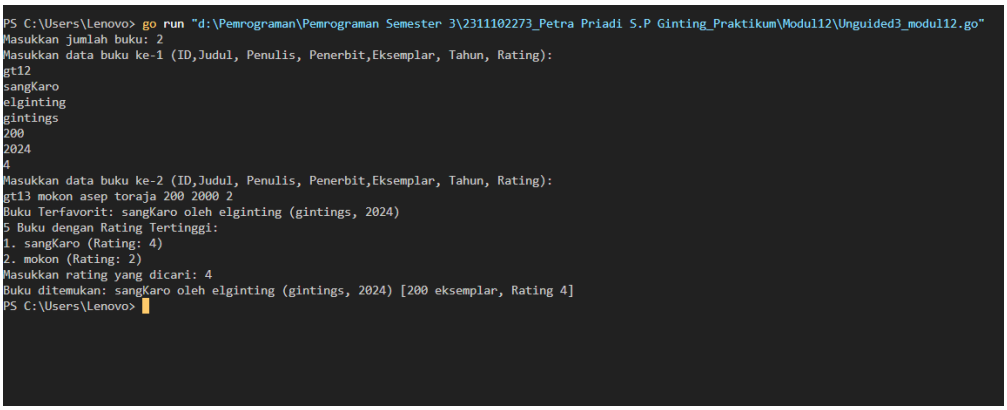
Sourcecode

```
package main

import "fmt"

const nMax = 7919

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

type DaftarBuku [nMax]Buku

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

func CetakTerfavorit(pustaka *DaftarBuku, n int) {
     if n == 0 {
           fmt.Println("Tidak ada buku dalam pustaka.")
           return
     }
     terbaik := pustaka[0]
     for i := 1; i < n; i++ {
           if pustaka[i].Rating > terbaik.Rating {
                 terbaik = pustaka[i]
           }
     }
     fmt.Printf("Buku Terfavorit: %s oleh %s (%s, %d)\n",
terbaik.Judul, terbaik.Penulis, terbaik.Penerbit,
terbaik.Tahun)
}

func UrutBuku(pustaka *DaftarBuku, n int) {
     for i := 1; i < n; i++ {
```

```
            key := pustaka[i]
            j := i - 1
            for j >= 0 && pustaka[j].Rating < key.Rating {
                  pustaka[j+1] = pustaka[j]
                  j--
            }
            pustaka[j+1] = key
      }
}

func Cetak5Terbaru(pustaka *DaftarBuku, n int) {
      fmt.Println("5 Buku dengan Rating Tertinggi: ")
      for i := 0; i < n && i < 5; i++ {
            fmt.Printf("%d. %s (Rating: %d)\n", i+1,
pustaka[i].Judul, pustaka[i].Rating)
      }
}

func CariBuku(pustaka *DaftarBuku, n, rating int) {
      left, right := 0, n-1
      for left <= right {
            mid := (left + right) / 2
            if pustaka[mid].Rating == rating {
                  b := pustaka[mid]
                  fmt.Printf("Buku ditemukan: %s oleh %s
(%s, %d) [%d eksemplar, Rating %d]\n",
                        b.Judul, b.Penulis, b.Penerbit,
b.Tahun, b.Eksemplar, b.Rating)
                  return
            } else if pustaka[mid].Rating < rating {
                  right = mid - 1
            } else {
                  left = mid + 1
            }
      }
      fmt.Println("Tidak ada buku dengan rating seperti
itu")
}

func main() {
      var pustaka DaftarBuku
      var n, rating int

      fmt.Print("Masukkan jumlah buku: ")
      fmt.Scan(&n)

      DaftarkanBuku(&pustaka, n)

      CetakTerfavorit(&pustaka, n)
```

```
        UrutBuku(&pustaka, n)
        Cetak5Terbaru(&pustaka, n)

        fmt.Print("Masukkan rating yang dicari: ")
        fmt.Scan(&rating)
        CariBuku(&pustaka, n, rating)
    }
```

Screenshoot Output



Deskripsi Program

Program dimulai dengan meminta jumlah buku yang akan didaftarkan, lalu pengguna memasukkan detail setiap buku, termasuk ID, judul, penulis, penerbit, jumlah eksemplar, tahun, dan rating. Fungsi CetakTerfavorit menemukan buku dengan rating tertinggi, sedangkan fungsi UrutBuku mengurutkan daftar buku dalam urutan menurun berdasarkan rating menggunakan algoritma Insertion Sort. Fungsi Cetak5Terbaru menampilkan hingga lima buku dengan rating tertinggi dari daftar yang sudah diurutkan. Selain itu, fungsi CariBuku menggunakan pencarian biner untuk mencari buku dengan rating tertentu di daftar yang sudah diurutkan. Program ini memastikan bahwa data buku dapat dikelola secara efisien, dengan kemampuan menampilkan informasi penting seperti buku terfavorit dan buku dengan rating tertinggi, serta mencari buku secara cepat.