LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL XII & XIII

Pengurutan Data

Disusun Oleh :

Nadhif Atha Zaki / 2311102007

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 algoritma pengurutan yang bekerja dengan cara mencari elemen terkecil (atau terbesar, tergantung pada urutan yang diinginkan) dalam array dan menukarnya dengan elemen yang berada di posisi saat ini. Proses ini diulang dengan mempertimbangkan bagian array yang belum terurut, hingga seluruh array terurut. Algoritma ini memiliki waktu kompleksitas $O(n^2)$, yang menjadikannya kurang efisien untuk dataset besar, tetapi mudah dipahami dan diimplementasikan.

Insertion Sort adalah algoritma pengurutan yang membangun urutan elemen satu per satu dengan cara memindahkan elemen yang belum terurut ke posisi yang sesuai pada bagian yang sudah terurut. Dimulai dengan menganggap elemen pertama sudah terurut, kemudian elemen berikutnya disisipkan ke dalam urutan yang sudah ada dengan memindahkan elemen-elemen yang lebih besar ke kanan. Insertion Sort memiliki waktu kompleksitas $O(n^2)$ pada kasus terburuk, tetapi lebih efisien daripada Selection Sort pada dataset yang sudah hampir terurut.

II. GUIDED

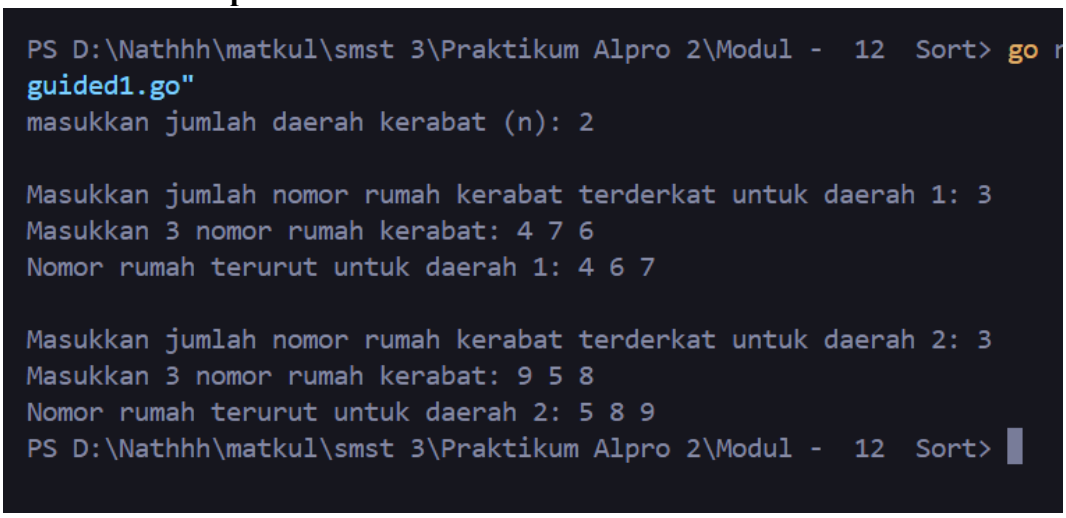
1.

Sourcecode

```
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()

    }

}
```

Screenshoot Output



Deskripsi Program

Program ini dirancang untuk mengurutkan nomor rumah kerabat di beberapa daerah menggunakan algoritma *selection sort*. Pengguna diminta untuk memasukkan jumlah daerah dan jumlah nomor rumah kerabat di setiap daerah. Nomor rumah kemudian diurutkan dari yang terkecil ke terbesar untuk setiap daerah secara terpisah. Setelah proses pengurutan selesai, program menampilkan hasil nomor rumah yang sudah terurut untuk masing-masing daerah. Program ini membantu menyusun data secara rapi dan mempermudah analisis atau pengelolaan informasi rumah kerabat di berbagai wilayah.

2.

Sourcecode

```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:\Nathhh\matkul\smst 3\Praktikum Alpro 2\Modul -  12  Sort> go run
guided2.go"
Masukkan data integer (akhiri dengan bilangan negatif):
4 5 3 2 -2
Array setelah diurutkan:
2 3 4 5
Data berjarak 1
PS D:\Nathhh\matkul\smst 3\Praktikum Alpro 2\Modul -  12  Sort>
```

Deskripsi Program

Program ini bertujuan untuk mengelola dan menganalisis data integer dengan fitur pengurutan dan pemeriksaan pola selisih. Pengguna dapat memasukkan data integer secara berurutan hingga menemukan bilangan negatif untuk mengakhiri input. Setelah data dimasukkan, program mengurutkannya menggunakan algoritma *insertion sort*. Kemudian, program memeriksa apakah selisih antara elemen-elemen yang telah diurutkan bersifat konstan, dan jika ya, menampilkan nilai selisih tersebut. Program ini memberikan keluaran berupa data yang sudah diurutkan dan informasi apakah data memiliki pola jarak tetap atau tidak.

III. UNGUIDED

1.

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"
    "sort"
)

// Struct untuk menyimpan kelompok nomor rumah
type KelompokRumah struct {
    nomorGanjil []int
    nomorGenap  []int
}

// Fungsi untuk mengelompokkan dan mengurutkan nomor rumah
func kelolaNomorRumah(nomorRumah []int) KelompokRumah {
    var kelompok KelompokRumah

    // Memisahkan nomor rumah menjadi ganjil dan genap
    for _, nomor := range nomorRumah {
        if nomor%2 == 0 {
            kelompok.nomorGenap =
append(kelompok.nomorGenap, nomor)
        } else {
```

```
            kelompok.nomorGanjil =
append(kelompok.nomorGanjil, nomor)
        }
    }

    // Mengurutkan dengan package sort
    sort.Ints(kelompok.nomorGanjil)
    // Urut naik untuk ganjil
    sort.Sort(sort.Reverse(sort.IntSlice(kelompok.nomorGenap
))) // Urut turun untuk genap

    return kelompok
}

func main() {
    var totalWilayah int

    // Meminta input jumlah wilayah
    fmt.Print("Masukkan jumlah wilayah: ")
    fmt.Scanln(&totalWilayah)

    // Pemrosesan setiap wilayah
    for wilayah := 1; wilayah <= totalWilayah; wilayah++ {
        var totalRumah int

        fmt.Printf("\nJumlah rumah di wilayah %d: ",
wilayah)
        fmt.Scanln(&totalRumah)

        // Membuat slice untuk menyimpan nomor rumah
        daftarRumah := make([]int, totalRumah)
        fmt.Printf("Masukkan %d nomor rumah: ", totalRumah)

        // Input nomor rumah
        for i := 0; i < totalRumah; i++ {
            fmt.Scan(&daftarRumah[i])
        }

        // Memproses pengelompokan dan pengurutan
        kelompokRumah := kelolaNomorRumah(daftarRumah)

        // Menampilkan hasil pengurutan
        fmt.Printf("\nNomor rumah terurut wilayah %d:\n",
wilayah)
        fmt.Print("Ganjil: ")
```

```
        for _, nomor := range kelompokRumah.nomorGanjil {
            fmt.Printf("%d ", nomor)
        }
        fmt.Print("\nGenap: ")
        for _, nomor := range kelompokRumah.nomorGenap {
            fmt.Printf("%d ", nomor)
        }
        fmt.Println()
    }
}
```

Screenshoot Output

```
PS D:\Nathhh\matkul\smst 3\Praktikum Alpro 2\Modul -  12  Sort> go run
1\unguided1.go"
Masukkan jumlah wilayah: 1

Jumlah rumah di wilayah 1: 5
Masukkan 5 nomor rumah: 2 3 4 8 4

Nomor rumah terurut wilayah 1:
Ganjil: 3
Genap: 8 4 4 2
PS D:\Nathhh\matkul\smst 3\Praktikum Alpro 2\Modul -  12  Sort>
```

Deskripsi Program

Program ini digunakan untuk mengelompokkan dan mengurutkan nomor rumah berdasarkan sifat ganjil dan genap di beberapa wilayah. Pengguna diminta memasukkan jumlah wilayah dan jumlah rumah di masing-masing wilayah, serta daftar nomor rumah. Nomor rumah ganjil diurutkan secara naik, sedangkan nomor rumah genap diurutkan secara turun menggunakan *package sort*. Hasil pengelompokan dan pengurutan ditampilkan secara terpisah untuk setiap wilayah dalam format yang rapi. Program ini membantu menyusun data nomor rumah dengan pola yang terorganisir, mempermudah pengguna dalam analisis dan pencatatan.

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 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.

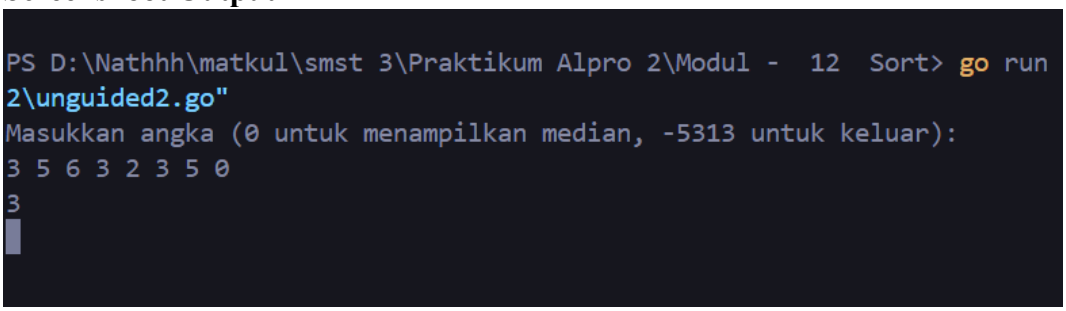
No	Masukan	Keluaran
1	7 23 11 0 5 19 2 29 3 13 17 0 -5313	11 12

Sourcecode

```go
package main

import (
    "fmt"
)

// Fungsi untuk mengurutkan array menggunakan insertion sort
func urutkanDenganInsertion(data []int) {
    for i := 1; i < len(data); i++ {
        elemen := data[i]
        j := i - 1
        for j >= 0 && data[j] > elemen {
            data[j+1] = data[j]
            j--
        }
        data[j+1] = elemen
    }
}
```

```go
// Fungsi untuk menghitung median dari array yang sudah terurut
func cariMedian(data []int) float64 {
    ukuran := len(data)
    if ukuran == 0 {
        return 0
    }

    // Jika jumlah elemen ganjil
    if ukuran%2 != 0 {
        return float64(data[ukuran/2])
    }

    // Jika jumlah elemen genap
    tengah := ukuran / 2
    return float64(data[tengah-1]+data[tengah]) / 2.0
}

func main() {
    var angka int
    nilai := []int{}

    fmt.Println("Masukkan angka (0 untuk menampilkan median, -5313 untuk keluar):")
    for {
        fmt.Scan(&angka)

        if angka == 0 {
            if len(nilai) > 0 {
                // Salin dan urutkan data sebelum menghitung median
                salinan := append([]int{}, nilai...)
                urutkanDenganInsertion(salinan)
                median := cariMedian(salinan)
                fmt.Printf("%.0f\n", median)
            }
        } else if angka == -5313 {
            // Keluar dari program
            break
        } else {
            // Tambahkan angka ke daftar nilai
            nilai = append(nilai, angka)
        }
    }
}
```

Screenshoot Output



Deskripsi Program

Program ini digunakan untuk menghitung median dari serangkaian angka yang dimasukkan oleh pengguna secara interaktif. Pengguna dapat terus memasukkan angka hingga angka 0 dimasukkan, yang akan memicu penghitungan dan penampilan median berdasarkan data yang sudah diurutkan menggunakan metode *insertion sort*. Jika pengguna memasukkan angka -5313, program akan berhenti. Program secara otomatis mengurutkan data yang telah dimasukkan sebelum menghitung median, baik untuk jumlah elemen ganjil maupun genap. Dengan fitur ini, program mempermudah analisis data numerik secara real-time.

3.

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

```go
package main

import (
    "fmt"
)

const maxBuku = 7919

// Struct untuk data buku
type Buku struct {
    id        string
    judul     string
    penulis   string
    penerbit  string
    eksemplar int
    tahun     int
    rating    int
}

// Struct untuk daftar buku
type KoleksiBuku []Buku
```

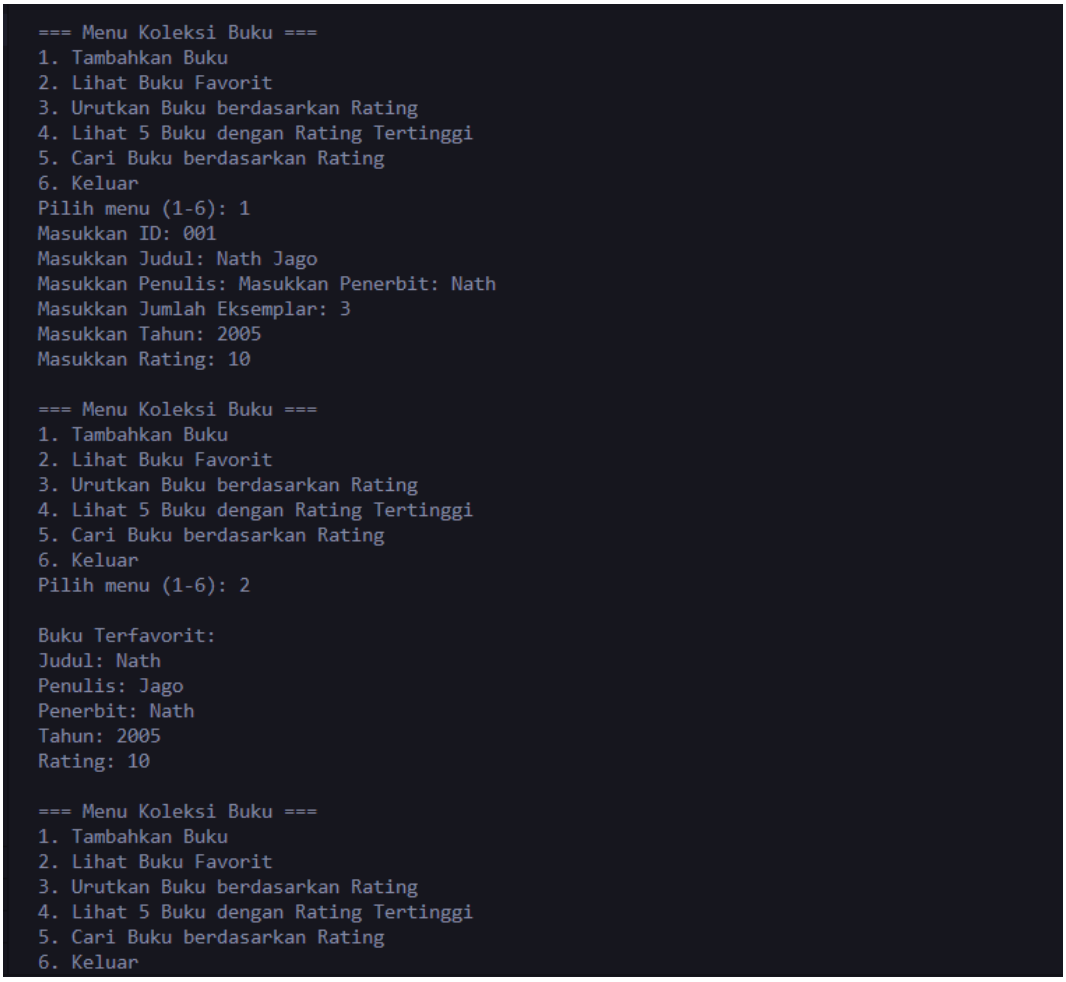
```go
// Fungsi untuk menambahkan buku baru
func TambahkanBuku(koleksi *KoleksiBuku, jumlah *int) {
    var bukuBaru Buku
    fmt.Print("Masukkan ID: ")
    fmt.Scan(&bukuBaru.id)
    fmt.Print("Masukkan Judul: ")
    fmt.Scan(&bukuBaru.judul)
    fmt.Print("Masukkan Penulis: ")
    fmt.Scan(&bukuBaru.penulis)
    fmt.Print("Masukkan Penerbit: ")
    fmt.Scan(&bukuBaru.penerbit)
    fmt.Print("Masukkan Jumlah Eksemplar: ")
    fmt.Scan(&bukuBaru.eksemplar)
    fmt.Print("Masukkan Tahun: ")
    fmt.Scan(&bukuBaru.tahun)
    fmt.Print("Masukkan Rating: ")
    fmt.Scan(&bukuBaru.rating)

    *koleksi = append(*koleksi, bukuBaru)
    *jumlah++
}

// Fungsi untuk mencetak buku dengan rating tertinggi
func TampilkanFavorit(koleksi KoleksiBuku, jumlah int) {
    if jumlah == 0 {
        fmt.Println("Tidak ada buku dalam koleksi")
        return
    }

    ratingTertinggi := koleksi[0].rating
    indeksFavorit := 0
    for i := 1; i < jumlah; i++ {
        if koleksi[i].rating > ratingTertinggi {
            ratingTertinggi = koleksi[i].rating
            indeksFavorit = i
        }
    }

    fmt.Println("\nBuku Terfavorit:")
    fmt.Printf("Judul: %s\nPenulis: %s\nPenerbit: %s\nTahun:
%d\nRating: %d\n",
        koleksi[indeksFavorit].judul,
koleksi[indeksFavorit].penulis,
        koleksi[indeksFavorit].penerbit,
koleksi[indeksFavorit].tahun,
```

```
        koleksi[indeksFavorit].rating)
}

// Fungsi untuk mengurutkan buku berdasarkan rating
menggunakan insertion sort
func SortirBuku(koleksi *KoleksiBuku, jumlah int) {
    for i := 1; i < jumlah; i++ {
        bukuSementara := (*koleksi)[i]
        j := i - 1
        for j >= 0 && (*koleksi)[j].rating <
bukuSementara.rating {
            (*koleksi)[j+1] = (*koleksi)[j]
            j--
        }
        (*koleksi)[j+1] = bukuSementara
    }
}

// Fungsi untuk menampilkan 5 buku dengan rating tertinggi
func CetakTop5(koleksi KoleksiBuku, jumlah int) {
    fmt.Println("\n5 Buku dengan Rating Tertinggi:")
    jumlahTampil := 5
    if jumlah < 5 {
        jumlahTampil = jumlah
    }
    for i := 0; i < jumlahTampil; i++ {
        fmt.Printf("%d. %s (Rating: %d)\n", i+1,
koleksi[i].judul, koleksi[i].rating)
    }
}

// Fungsi untuk mencari buku dengan rating tertentu
menggunakan binary search
func TemukanBuku(koleksi KoleksiBuku, jumlah int, target
int) {
    kiri := 0
    kanan := jumlah - 1
    ditemukan := false

    for kiri <= kanan {
        tengah := (kiri + kanan) / 2
        if koleksi[tengah].rating == target {
            fmt.Printf("\nBuku dengan rating %d
ditemukan:\n", target)
```

```
            fmt.Printf("Judul: %s\nPenulis: %s\nPenerbit:
%s\nTahun: %d\nEksemplar: %d\n",
                koleksi[tengah].judul,
koleksi[tengah].penulis,
                koleksi[tengah].penerbit,
koleksi[tengah].tahun,
                koleksi[tengah].eksemplar)
            ditemukan = true
            break
        }
        if koleksi[tengah].rating < target {
            kanan = tengah - 1
        } else {
            kiri = tengah + 1
        }
    }

    if !ditemukan {
        fmt.Printf("\nTidak ada buku dengan rating %d\n",
target)
    }
}

func main() {
    var koleksi KoleksiBuku
    var jumlahBuku int
    var pilihan int
    var ratingTarget int

    for {
        fmt.Println("\n=== Menu Koleksi Buku ===")
        fmt.Println("1. Tambahkan Buku")
        fmt.Println("2. Lihat Buku Favorit")
        fmt.Println("3. Urutkan Buku berdasarkan Rating")
        fmt.Println("4. Lihat 5 Buku dengan Rating
Tertinggi")
        fmt.Println("5. Cari Buku berdasarkan Rating")
        fmt.Println("6. Keluar")
        fmt.Print("Pilih menu (1-6): ")
        fmt.Scan(&pilihan)

        switch pilihan {
        case 1:
            TambahkanBuku(&koleksi, &jumlahBuku)
        case 2:
```

```
            TampilkanFavorit(koleksi, jumlahBuku)
        case 3:
            SortirBuku(&koleksi, jumlahBuku)
            fmt.Println("Buku berhasil diurutkan berdasarkan
rating")
        case 4:
            CetakTop5(koleksi, jumlahBuku)
        case 5:
            fmt.Print("Masukkan rating yang ingin dicari: ")
            fmt.Scan(&ratingTarget)
            TemukanBuku(koleksi, jumlahBuku, ratingTarget)
        case 6:
            fmt.Println("Terima kasih telah menggunakan
program koleksi buku.")
            return
        default:
            fmt.Println("Pilihan tidak valid. Coba lagi.")
        }
    }
}
```

Screenshoot Output



Deskripsi Program

Program ini berfungsi untuk mengelola koleksi buku dengan berbagai fitur, seperti menambahkan buku, mengurutkan buku berdasarkan rating, dan mencari buku berdasarkan rating. Pengguna dapat menambahkan buku baru dengan memasukkan informasi seperti ID, judul, penulis, penerbit, jumlah eksemplar, tahun, dan rating buku. Fitur lainnya termasuk menampilkan buku dengan rating tertinggi, mengurutkan buku menggunakan metode Insertion Sort, serta menampilkan lima buku teratas berdasarkan rating. Program juga dilengkapi dengan kemampuan pencarian buku berdasarkan rating menggunakan metode Binary Search. Antarmuka pengguna yang sederhana memungkinkan pengoperasian yang mudah dan interaktif untuk mengelola koleksi buku.