

**LAPORAN PRAKTIKUM  
ALGORITMA PEMOGRAMAN 2**

**MODUL 11**

**MATERI**

**PENGURUTAN DATA**



Oleh:

**FEBRIAN FALIH ALWAFI**

**2311102181**

**S1F-11-02**

**S1 TEKNIK INFORMATIKA**

**UNIVERSITAS TELKOM PURWOKERTO**

**2024**

## **I. DASAR TEORI**

- **Ide Algoritma 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) Pindahkan/tukar tempat dengan data yang berada pada posisi paling kiri pada rentang data tersisa tersebut.
- 3) Ulangi 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 swop.

- **Algoritma selection sort**

Adapun algoritma selection sort pada untuk mengurutkan array bertipe data bilangan bulat secara. Secara membesar atau ascending.

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

```

- **Ide Algoritma Insertion Sort**

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 data dengan indeks kecil ada di "kiri" dan indeks besar ada di "kanan".

1) Untuk satu data yang belum terurut dan sejumlah data yang sudah diurutkan:

Geser data yang sudah terurut tersebut (ke kanan), sehingga ada suatu ruang kosong untuk memasukkan data yang belum terurut ke dalam data yang sudah terurut dan tetap menjaga keterurutan.

2) Ulangi proses tersebut untuk setiap data yang belum terurut terhadap rangkaian data yang sudah terurut.

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

- **Algoritma Insertion Sort**

Adapun algoritma insertion sort pada untuk mengurutkan array berisi data bilangan bulat secara menaik atau descending adalah sebagai berikut ini!

```
...
type arrInt [4321]int
...
func insertionSort1(T *arrInt, n int){
/* I.S. terdefinisi array T yang berisi n bilangan bulat
   F.S. array T terurut secara mengecil atau descending dengan INSERTION SORT*/
var temp, i, j int
i = 1
for i <= n-1 {
    j = i
    temp = T[j]
    for j > 0 && temp > T[j-1] {
        T[j] = T[j-1]
        j = j - 1
    }
    T[j] = temp
    i = i + 1
}
}
```

Sama halnya apabila array yang akan diurutkan adalah bertipe data struct, maka tambahkan field pada saat proses perbandingan dalam pencarian posisi, kemudian tipe data dari variabel temp sama dengan struct dari arraynya.

## II. GUIDED

### 1. 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
        }

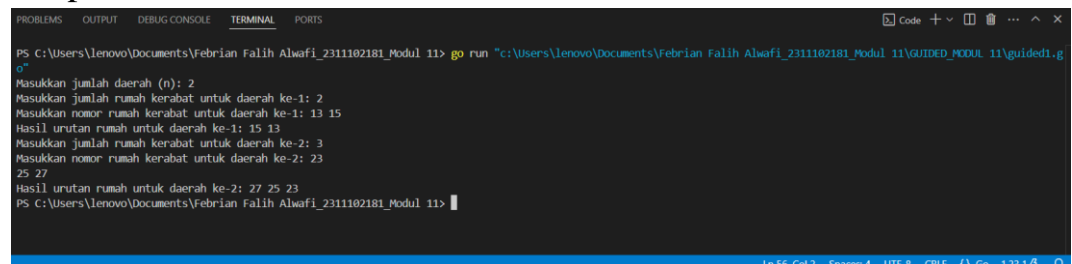
        // Masukkan nomor rumah
        houses := make([]int, m)
        fmt.Printf("Masukkan nomor rumah
kerabat untuk daerah ke-%d: ", i+1)
        for j := 0; j < m; j++ {
            fmt.Scan(&houses[j])
        }

        // 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:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 11> go run "c:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 11\GUIDED_MODULE_11\guided1.g
o"
Masukkan jumlah daerah (n): 2
Masukkan jumlah rumah kerabat untuk daerah ke-1: 2
Masukkan nomor rumah kerabat untuk daerah ke-1: 13 15
Hasil urutan rumah untuk daerah ke-1: 13 15
Masukkan jumlah rumah kerabat untuk daerah ke-2: 3
Masukkan nomor rumah kerabat untuk daerah ke-2: 23
25 27
Hasil urutan rumah untuk daerah ke-2: 23 25 27
PS C:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 11>

```

### Deskripsi Program :

Program di atas digunakan untuk mengurutkan nomor rumah kerabat di beberapa daerah menggunakan algoritma Selection Sort. Program ini dimulai dengan meminta pengguna memasukkan jumlah daerah (n). Nilai n harus berada dalam rentang 1 hingga 999 untuk memastikan input valid. Untuk setiap daerah, pengguna diminta untuk memasukkan jumlah rumah kerabat (m), yang juga harus berada dalam rentang 1 hingga 999,999. Jika ada input yang tidak valid, program akan menampilkan pesan kesalahan dan langsung berhenti.

## 2. GUIDED 2

Source Code :

```
package main

import (
    "fmt"
    "math"
)

// Fungsi insertion sort untuk mengurutkan array
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:\Users\lenovo\Documents\Febrian Falih Alwafi_2311182181_Modul 11> go run "c:\Users\lenovo\Documents\Febrian Falih Alwafi_2311182181_Modul 11\GJIDED_MODUL 11\guided2.g
o"
Masukkan data (akhiri dengan bilangan negatif):
18 2 10 34 26 42 58 50 -2
Hasil pengurutan: [2 10 18 26 34 42 50 58]
Data berjarak 8
PS C:\Users\lenovo\Documents\Febrian Falih Alwafi_2311182181_Modul 11>

```

### Deskripsi Program :

Program ini mengurutkan array bilangan dan memeriksa apakah elemen-elemen dalam array tersebut memiliki jarak yang tetap antara satu dengan lainnya. Program ini menggunakan algoritma **Insertion Sort** untuk mengurutkan data dan menghitung selisih antara elemen-elemen dalam array untuk memeriksa konsistensi jaraknya. Sebagai contoh, jika pengguna memasukkan data 18 2 10 34 26 42 58 50 -2, program akan mengurutkan data menjadi [2, 10, 18, 26, 34, 42, 50, 58]. Program kemudian akan memeriksa jarak antara elemen-elemen array dan mencetak Data berjarak 8, karena selisih antara setiap elemen adalah 8. Jika selisih antar elemen tidak sama, program akan mencetak Data berjarak tidak tetap.

## III. UNGUIDED

### 1. UNGUIDED 1

Source Code :

```
package main

import (
    "fmt"
    "strings"
)

func sortAscending(arr []int) {
    for i := 0; i < len(arr)-1; i++ {
        for j := i + 1; j < len(arr); j++ {
```

```

        if arr[j] < arr[i] {
            arr[i], arr[j] = arr[j], arr[i]
        }
    }
}

func sortDescending(arr []int) {
    for i := 0; i < len(arr)-1; i++ {
        for j := i + 1; j < len(arr); j++ {
            if arr[j] > arr[i] {
                arr[i], arr[j] = arr[j], arr[i]
            }
        }
    }
}

func processHouses_2311102181(houses []int)
(string, string) {
    var ganjil, genap []int

    for _, num := range houses {
        if num%2 == 0 {
            genap = append(genap, num)
        } else {
            ganjil = append(ganjil, num)
        }
    }

    sortDescending(ganjil)
    sortAscending(genap)

    ganjilStr := strings.Trim(fmt.Sprint(ganjil),
"[]")
    genapStr := strings.Trim(fmt.Sprint(genap),
"[]")

    return ganjilStr, genapStr
}

```

```

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

    if n <= 0 {
        fmt.Println("Jumlah daerah harus lebih
        besar dari 0.")
        return
    }

    results := make([]string, 0, n)

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

        if m <= 0 {
            fmt.Println("Jumlah rumah kerabat harus
            lebih besar dari 0.")
            continue
        }

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

        ganjilStr, genapStr :=
        processHouses_2311102181(houses)
        results = append(results,
        fmt.Sprintf("%s\n%s", ganjilStr, genapStr))
    }
}

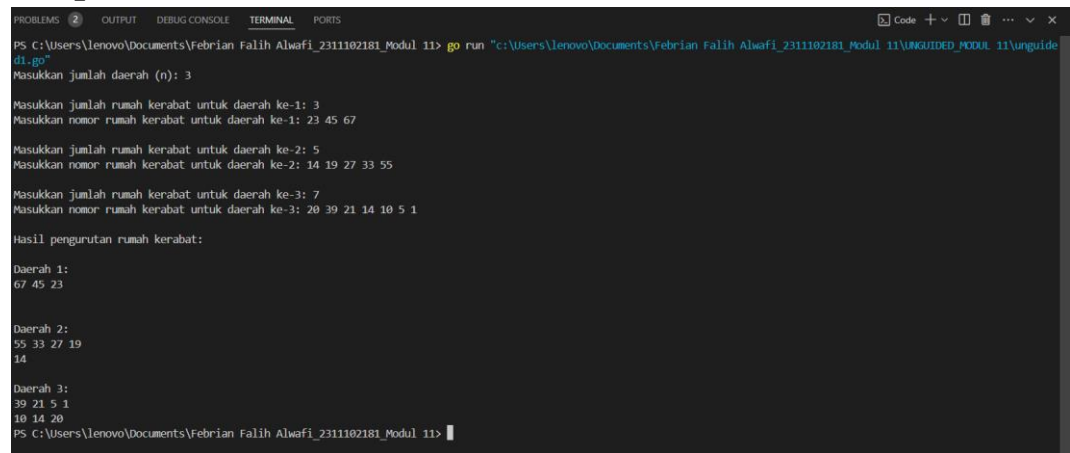
```

```

        fmt.Println("\nHasil pengurutan rumah
kerabat:")
        for i, result := range results {
            fmt.Printf("\nDaerah %d:\n%s\n", i+1,
result)
        }
    }
}

```

## Output :



```

PS C:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 11> go run "C:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 11\unguide
di.go"
Masukkan jumlah daerah (n): 3
Masukkan jumlah rumah kerabat untuk daerah ke-1: 3
Masukkan nomor rumah kerabat untuk daerah ke-1: 23 45 67
Masukkan jumlah rumah kerabat untuk daerah ke-2: 5
Masukkan nomor rumah kerabat untuk daerah ke-2: 14 19 27 33 55
Masukkan jumlah rumah kerabat untuk daerah ke-3: 7
Masukkan nomor rumah kerabat untuk daerah ke-3: 20 39 21 14 10 5 1
Hasil pengurutan rumah kerabat:
Daerah 1:
67 45 23
Daerah 2:
55 33 27 19
14
Daerah 3:
39 21 5 1
10 14 20
PS C:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 11>

```

## Deskripsi Program :

Program diatas di gunakan untuk membantu mengelompokkan nomor rumah kerabat berdasarkan **ganjil** dan **genap** di beberapa daerah. Setelah pengelompokan, nomor rumah ganjil diurutkan secara **menurun** (descending), sedangkan nomor rumah genap diurutkan secara **menaik** (ascending). Program ini menerima input dari pengguna dan menampilkan hasilnya dalam format yang terstruktur untuk setiap daerah. Program ini dimulai dengan mendeklarasikan dua fungsi, yaitu **sortAscending** dan **sortDescending**, yang masing-masing bertugas mengurutkan elemen array secara menaik dan menurun. Fungsi-fungsi ini menggunakan algoritma

**bubble sort** sederhana, yang membandingkan setiap pasangan elemen dalam array dan menukar posisinya jika syarat pengurutan tidak terpenuhi. Sebagai contoh jika kamu memasukkan jumlah daerah 3 maka kamu akan memasukkan jumlah rumah kerabat sampai daerah ke 3 seperti pada output diatas.

## 2. UNGUIDED 2

Source Code :

```
package main

import (
    "fmt"
    "sort"
)

func calculateMedian_2311102181(arr []int) int
{
    n := len(arr)
    if n%2 == 1 {
        return arr[n/2]
    }
    return (arr[n/2-1] + arr[n/2]) / 2
}

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

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

        if input == -5313 {
            break
        }
    }
}
```

```

    }

    if input == 0 {

        if len(data) == 0 {
            fmt.Println("Data kosong, tidak dapat
menghitung median.")
            continue
        }

        sort.Ints(data)
        median :=
calculateMedian_2311102181(data)
        fmt.Println("Median:", median)
    } else {

        data = append(data, input)
    }
}
}

```

Output :

```

PS C:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 11> go run "c:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 11\unguide
d2.go"
Masukkan angka (0 untuk menghitung median, -5313 untuk keluar):
7 23 11 0 5 19 2 29 3 13 17 0 -5313
Median: 11
Median: 12
PS C:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 11>

```

Deskripsi Program :

Program yang menerima input angka dari pengguna, mengurutkan angka-angka tersebut, dan menghitung median setiap kali pengguna memasukkan angka 0. Program akan terus menerima input angka hingga

pengguna memasukkan angka -5313, yang akan menghentikan program. Fungsi utama dalam kode ini adalah menghitung median dari serangkaian angka yang telah dimasukkan, dengan ketentuan angka-angka tersebut harus diurutkan terlebih dahulu. Program ini juga memanfaatkan loop untuk terus menerima input dan mengelola data dengan efisien, sekaligus memberikan respons yang jelas saat input tidak valid atau ketika tidak ada data untuk dihitung median-nya.

### 3. UNGUIDED 3

Source Code :

```
package main

import "fmt"

const nMax = 7919

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

type DaftarBuku struct {
    Pustaka_2311102181 [nMax]Buku
    nPustaka int
}

func sortBukuByRating(pustaka *DaftarBuku) {

    for i := 1; i < pustaka.nPustaka; i++ {
        key := pustaka.Pustaka_2311102181[i]
        j := i - 1
```



```

        for j >= 0 &&
        pustaka.Pustaka_2311102181[j].Ranting <
        key.Ranting {
            pustaka.Pustaka_2311102181[j+1] =
            pustaka.Pustaka_2311102181[j]
            j--
        }
        pustaka.Pustaka_2311102181[j+1] = key
    }
}

func printFavoritBuku(pustaka DaftarBuku) {
    if pustaka.nPustaka == 0 {
        fmt.Println("Tidak ada buku.")
        return
    }

    favorit := pustaka.Pustaka_2311102181[0]
    for _, b := range
    pustaka.Pustaka_2311102181[:pustaka.nPustaka]
    {
        if b.Ranting > favorit.Ranting {
            favorit = b
        }
    }
    fmt.Printf("Buku Terfavorit: %s %s %s %s
    Tahun: %d\n", favorit.ID, favorit.Judul,
    favorit.Penulis, favorit.Penerbit, favorit.Tahun)
}

func printTop5Buku(pustaka DaftarBuku) {
    if pustaka.nPustaka < 5 {
        fmt.Println("Jumlah buku kurang dari 5,
        menampilkan semua buku.")
    }
    fmt.Println("5 Buku dengan Rating
    Tertinggi:")

```

```

        for i := 0; i < pustaka.nPustaka && i < 5; i++
        {
            fmt.Printf("%s ",
pustaka.Pustaka_2311102181[i].Judul)
        }
        fmt.Println()
    }

func searchBukuByRating(pustaka DaftarBuku,
rating int) *Buku {
    low, high := 0, pustaka.nPustaka-1
    for low <= high {
        mid := (low + high) / 2
        if
pustaka.Pustaka_2311102181[mid].Ranting ==
rating {
            return
&pustaka.Pustaka_2311102181[mid]
        } else if
pustaka.Pustaka_2311102181[mid].Ranting <
rating {
            low = mid + 1
        } else {
            high = mid - 1
        }
    }
    return nil
}

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

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

    fmt.Println("Masukkan data buku (ID, Judul,
Penulis, Penerbit, Eksemplar, Tahun, Rating):")
    for i := 0; i < n; i++ {

```

```

        var b Buku
        fmt.Print("Masukkan : ")
        fmt.Scan(&b.ID, &b.Judul, &b.Penulis,
        &b.Penerbit, &b.Eksemplar, &b.Tahun,
        &b.Ranting)
        pustaka.Pustaka_2311102181[i] = b
    }

    pustaka.nPustaka = n

    fmt.Print("Masukkan rating buku yang Anda
    cari: ")
    fmt.Scan(&cariRating)

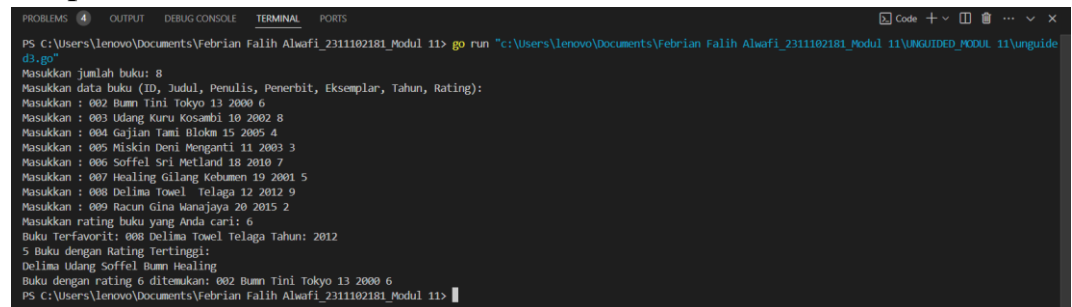
    sortBukuByRating(&pustaka)

    printFavoritBuku(pustaka)
    printTop5Buku(pustaka)

    temukan := searchBukuByRating(pustaka,
    cariRating)
    if temukan != nil {
        fmt.Printf("Buku dengan rating %d
        ditemukan: %s %s %s %s %d %d %d\n",
        cariRating, temukan.ID, temukan.Judul,
        temukan.Penulis, temukan.Penerbit,
        temukan.Eksemplar, temukan.Tahun,
        temukan.Ranting)
    } else {
        fmt.Printf("Buku dengan rating %d tidak
        ditemukan.\n", cariRating)
    }
}

```

Output :



```
PS C:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 11> go run "c:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 11\UNGUJIDED_MODAL 11\unguide
d3.go"
Masukkan jumlah buku: 8
Masukkan data buku (ID, Judul, Penulis, Penerbit, Eksemplar, Tahun, Rating):
Masukkan : 002 Bumi Tini Tokyo 13 2000 6
Masukkan : 003 Udang Kuru Kosambi 10 2002 8
Masukkan : 004 Gajian Tami Blokm 15 2005 4
Masukkan : 005 Miskin Deni Menganti 11 2003 3
Masukkan : 006 Soffel Sri Metland 18 2010 7
Masukkan : 007 Healing Gilang Kebumen 19 2001 5
Masukkan : 008 Delima Towel Telaga 12 2012 9
Masukkan : 009 Racun Gina Manajaya 20 2015 2
Masukkan rating buku yang Anda cari: 6
Buku terfavorit: 008 Delima Towel Telaga Tahun: 2012
5 Buku dengan Rating Tertinggi:
Delima Udang soffel Bumi Healing
Buku dengan rating 6 ditemukan: 002 Bumi Tini Tokyo 13 2000 6
PS C:\Users\lenovo\Documents\Febrian Falih Alwafi_2311102181_Modul 11> |
```

Deskripsi Program :

Program ini adalah manajemen koleksi buku yang berfokus pada pengurutan buku berdasarkan rating, pencarian buku berdasarkan rating, dan menampilkan informasi terkait buku favorit dan buku terbaik. Program ini dimulai dengan mendefinisikan struktur data yang menyimpan informasi buku, yaitu ID, judul, penulis, penerbit, jumlah eksemplar, tahun penerbitan, dan rating. Struktur DaftarBuku digunakan untuk menyimpan daftar buku yang dimasukkan oleh pengguna, dengan kapasitas maksimum hingga 7919 buku. Secara keseluruhan, program ini memberikan kemudahan dalam mengelola koleksi buku berdasarkan rating, dengan kemampuan untuk mencari, menyortir, dan menampilkan buku favorit serta buku dengan rating tertinggi, yang menjadikannya alat yang efektif dalam manajemen koleksi buku. Seperti contoh pada output diatas yang menampilkan hasil akhirnyaada Buku terfavorit, 5 buku dengan rating tertinggi, dan buku yang ingin kita cari dengan rating.