LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL XII, XIII

PENGURUTAN DATA

Disusun Oleh :

Deshan Rafif Alfarisi / 2311102326

S1-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

1. Algorltma Selectlon 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 (ascendinp), 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 *suJop.*

	Notasi Algoritma	Notasi datam bahasa Go
1	`i 1`	`i 1`
2	`while i ‹= n-1 do`	`for i ‹= n-1 {`
3	`idx min t 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`**	`}`
U	`t *- a[idx_min]`	`t a[idx min]`
12	`a[idx_min] +- a[ı-1 ]`	`a[idx min] a[i-1]`
13	`a[i-1 ] €- t`	`a[i-1 ] t`
14	`1 'E- i + 1`	`i i + 1`
15	`endwhile`	`}`

2. Algorltma Selectlon Sort

Adapun algoritma *selection* sort pada untuk mengurutkan array bertipe data bilangan bulat secara membesar atau ascending adalah sebagai berikut ini!

```
5   type arrlnt [4321 ]:înt
    func selectionsort1 (T *ar rInt, n int)(
15  /* Z.S. terdef1n1st array T yang ber1s1 n b11angan bulat
16     F.S. array T terurt/t secara ascedlng atau nenbesar dengan SELECTZ0fiI SORT */
17      var t, i, j, idx_min int
18
19
```

3. lde Aigorltma lnsertlon 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 satu 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 **/nsertfon Sort,** yang mana pada algoritma ini melibatkan dua proses yaitu pencarian sekuensial dan penyisipan.

	Notasl A1gor1toa	Notasl da1am bahasa Go
1	`i G 1`	`:î 1`
2	`while i <= n-1 do`	`lo i «= n-1 (`
	`j 'E- i`	`r`
3	`temp G a[j]`	`j i`
4	`whıle j > 0 and temp > a[`	`temp = a[j]`
5	`j -1 ] do a[ j ] +- a [`	`lo r j > 0 && temp > a[ j`
	`j -1 ]`	`-1 ] (`
6	`j +- j – 1`	`a[ j ] = a [ j -1 ]`
7	`endwhile`	`j j – 1`
8	`a[ j ] G`	
9	`terıp i`	`a[j] temp`
10	`'E- i + 1`	`i i + 1`
11	`endwhile`	`}`

4. Algorltma insertion Sort

Adapun algoritma *insertion sort* pada untuk mengurutkan array bertipe data bilangan bulat secara mengecil atau descending adalah sebagai berikut ini!

```
5
type arrInt [4327]int
15
func :fnsertionSort1 (T *arrInt, n int){
16
/^ Z.S. terdef1n1st array T yang ber1s1 n b1tangan but at
17
   F.S. array T terurut secara nengec11 atau descend1ng dengan ZIJSERTZ0iJ SORT*/
18
    var temp, i,   j int
19  i    1
20  for i «= n-1 (
21      j = i
22      temp = T[j]
23      for j » 0 && temp » T[j-1] {
24          T[j] = T[j-1]
25          j    j — 1
26      }
27      T[j]   temp
        i   i + 1
28  }
29
}
30
```


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.

```
type riahasiswa st ruct (
    nama, nim, kelas, jurusan string
    ipk  f1oat64
type arrMhs [2023]mahasiswa

func insertionsort2 (T * arrMhs,  n int){
/* Z.S. terde f1n1st  array T yang ber1s1 n data nahasls a
   F.S. array T terurut secara nengec11 atau descend1ng berdasarkan nana dengan
nenggunakan a1gor1t/ra ZNSERTZ0fiI SORT */
    va r  temp  i,    j  int
    var  temp  mahas1swa
    :f    1
    lo  r  :t  «= n-1  (
        j  = 1
        temp  = T[ j ]
        lo r  j  »  0  && temp.nama  »  T[ j -1 j .nama  (
            T[ j j  =  T[ j -1 j
            j    j  —  1
        }
        T[ j ]    temp
        :t    1  +  1
```

II. GUIDED

1. Hercules, preman terkenal seantero ibukota, memiliki kerabat di banyak daerah. Tentunya Hercules sangat suka mengunjungi semua kerabatnya itu.

   Diberikan masukan nomor rumah dari semua kerabatnya di suatu daerah, buatlah program rumahkerabat yang akan menyusun nomor-nomor rumah kerabatnya secara terurut membesar menggunakan algoritma selection sort.

   Masukan dimulai dengan sebuah integer n (0 ‹ n ‹ 1000), banyaknya daerah kerabat Hercules tinggal. Isi n baris berikutnya selalu dimulai dengan sebuah integer m (0 ‹ m ‹ 1000000) yang menyatakan banyaknya rumah kerabat di daerah tersebut, diikuti dengan rangkaian bilangan bulat positif, nomor rumah para kerabat.

   Keluaran terdiri dari n baris, yaitu rangkaian rumah kerabatnya terurut membesar di masing- masing daerah.

   Sourcecode

```go
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

```
Masukkan jumlah daerah kerabat (n): 1

Masukkan jumlah nomor rumah kerabat untuk daerah 1: 3
Masukkan 3 nomor rumah kerabat: 4
57
9
Nomor rumah terurut untuk daerah 1: 4 9 57
PS C:\Users\Lenovo\Documents\file kuliah\Laprak Alpro 2\Modul 12> go run "c:\Users\Leno
```

Deskripsi Program

Program ini mengurutkan nomor rumah kerabat dari berbagai daerah menggunakan algoritma Selection Sort. Pengguna akan diminta memasukkan jumlah daerah dan jumlah nomor rumah untuk setiap daerah. Kemudian, program akan membaca nomor-nomor rumah, mengurutkannya dari yang terkecil hingga terbesar menggunakan Selection Sort, dan menampilkan hasil pengurutan untuk setiap daerah. Selection Sort bekerja dengan cara mencari elemen terkecil dalam setiap iterasi dan menukarnya dengan elemen pertama yang belum diurutkan. Proses ini diulang hingga seluruh array terurut.

2 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.

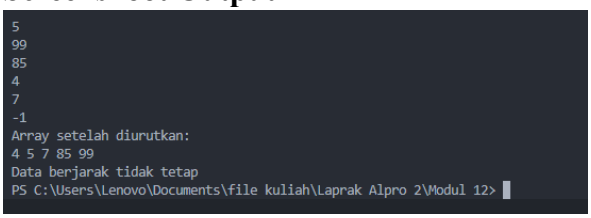
Keluaran terdiri dari dua baris. Baris pertama adalah isi dari array setelah dilakukan pengurutan, sedangkan baris kedua adalah status jarak setiap bilangan yang ada di dalam array."Data berjarak x" atau "data berjarak tidak tetap".

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



Deskripsi Program

Program ini mengurutkan serangkaian bilangan bulat yang diinput pengguna menggunakan algoritma Insertion Sort. Setelah pengurutan, program memeriksa apakah selisih antara setiap bilangan yang berdekatan dalam array yang telah terurut selalu sama (konstan). Jika selisihnya konstan, program akan menampilkan nilai selisih tersebut. Algoritma Insertion Sort bekerja dengan cara menyisipkan setiap elemen ke posisi yang benar dalam subarray yang telah terurut sebelumnya.

III. UNGUIDED

1. Soal Studi Case

Keterangan: Terdapat 3 daerah dalam contoh input, dan di masing-masing daerah mempunyai 5, 6, dan 3 kerabat.

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.

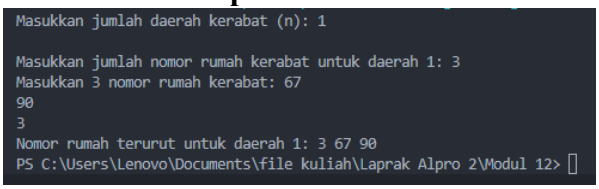
Format Masukan masih persis sama seperti sebelumnya.
Keluaran terdiri dari n baris, yaitu rangkaian rumah kerabatnya terurut membesar untuk nomor ganjil, diikuti dengan terurut mengecil untuk nomor genap, di masing-masing daerah.

Sourcecode

```go
package main

import "fmt"

const maxKelinci = 1000

func main() {
    var n int
```

```
    var berat [maxKelinci]float64
    var terkecil, terbesar float64

    // Meminta input jumlah kelinci
    fmt.Print("Masukkan jumlah anak kelinci: ")
    fmt.Scan(&n)

    // Validasi input
    if n <= 0 || n > maxKelinci {
        fmt.Println("Jumlah kelinci harus antara 1 dan",
maxKelinci)
        return
    }

    // Memasukkan data berat kelinci
    fmt.Println("Masukkan berat masing-masing kelinci (dalam
kg):")
    for i := 0; i < n; i++ {
        fmt.Scan(&berat[i])
    }

    // Inisialisasi nilai terkecil dan terbesar dengan berat
kelinci pertama
    terkecil = berat[0]
    terbesar = berat[0]

    // Mencari nilai terkecil dan terbesar
    for i := 1; i < n; i++ {
        if berat[i] < terkecil {
            terkecil = berat[i]
        }
        if berat[i] > terbesar {
            terbesar = berat[i]
        }
    }

    // Menampilkan hasil
    fmt.Printf("Berat kelinci terkecil: %.2f kg\n",
terkecil)
    fmt.Printf("Berat kelinci terbesar: %.2f kg\n",
terbesar)
}
```

Screenshoot Output



Deskripsi Program

Program ini mengurutkan nomor rumah kerabat dari berbagai daerah berdasarkan genap dan ganjil. Program ini menggunakan algoritma Selection Sort untuk mengurutkan nomor ganjil secara menaik dan nomor genap secara menurun. Setelah pengurutan, kedua kelompok nomor tersebut digabungkan dan ditampilkan. Dengan kata lain, program ini memodifikasi algoritma Selection Sort dasar untuk memenuhi kebutuhan pengurutan yang lebih spesifik, yaitu mengurutkan nomor berdasarkan paritasnya (ganjil atau genap) dan arah pengurutan yang berbeda untuk setiap kelompok.

2. Soal 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 untub menyelesaikannya, tetapi hanya 3 tim yang berhasil. Apa sih problemnya?

Median ada/ah nilai tengah dari *suatu boleksi* data yang suda/ı terurut. /ikajumÎa/ı data penap, maka ni/ai median adalah rerata dari kedua nilai tenga/ınya. *Pada* problem *ini,* semua data ”merupakan *bilangan* bulat *positif,* dan karenanya rerata *nilai* tenpo/ı dibu/atkan *İle 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 masuban diakhiri dengan bilangan bulat -5313.

Keluaran adalah median yang diminta, satu data per baris.

Sourcecode

```
package main

import "fmt"

const maxSize = 1000000

func sortArray(data2217 []int, ukuran int) {
    for start2 := 0; start2 < ukuran-1; start2++ {
        smallest := start2
        for check := start2 + 1; check < ukuran; check++ {
            if data2217[check] < data2217[smallest] {
                smallest = check
            }
        }
        data2217[start2], data2217[smallest] =
data2217[smallest], data2217[start2]
    }
}

func hitungMedian217(data []int, size int) float64 {
    mid := size / 2
    if size%2 == 0 {
        return float64(data[mid-1]+data[mid]) / 2.0
    }
    return float64(data[mid])
}

func main() {
    var values [maxSize]int
    var sorted [maxSize]int
    var count int
    var input int

    fmt.Println("Masukkan bilangan bulat (gunakan 0 untuk
menampilkan median sementara, akhiri dengan -5313):")
    for {
        fmt.Scan(&input)
        if input == -5313 {
            break
        }

        if input != 0 {
            values[count] = input
```

```
            count++
        } else if count > 0 {
            for i := 0; i < count; i++ {
                sorted[i] = values[i]
            }

            sortArray(sorted[:], count)
            median := hitungMedian217(sorted[:], count)
            fmt.Printf("%.0f\n", median)
        }
    }
}
```

Screenshoot Output

```
Masukkan bilangan bulat (gunakan 0 untuk menampilkan median sementara, akhiri dengan -5
313):
3 4 5 6 67  98 23 0 2 1 -5313
6
PS C:\Users\Lenovo\Documents\file kuliah\Laprak Alpro 2\Modul 12>
```

Deskripsi Program

Program ini dirancang untuk menghitung dan menampilkan median dari serangkaian bilangan bulat yang diinputkan oleh pengguna. Program ini menggunakan algoritma Selection Sort untuk mengurutkan data sebelum menghitung median. Pengguna dapat memasukkan bilangan bulat secara berurutan, dan setiap kali angka 0 dimasukkan, program akan menghitung dan menampilkan median dari bilangan-bilangan yang telah diinput sebelumnya. Setelah menampilkan median, program akan mengosongkan data yang telah diproses dan siap untuk menerima input bilangan berikutnya.

3. Soal Studi Case

Sebuah program perpustakaan digunakan untuk mengelola data buku di dalam suatu perpustakaan. Misalnya terdefinisi struct dan array seperti beribut ini:

IV.

```
const nMax : integer = 7919
type Buku = <

    id, judul, penulis, penerbit : string
    eksemplar, tahun, rating : integer »

type DaftarBuku = array [ 7..nMax] of Buku
```

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.

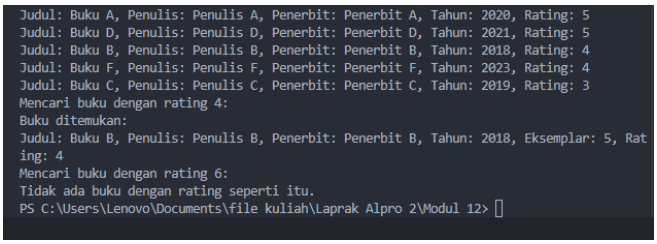
Sourcecode

```
package main

import (
    "fmt"
)

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

// Daftar Buku
type DaftarBuku []Buku

// Fungsi untuk mengurutkan buku berdasarkan rating secara
descending menggunakan insertion sort
func UrutBuku(pustaka DaftarBuku) DaftarBuku {
    n := len(pustaka)
    for i := 1; i < n; i++ {
        key := pustaka[i]
        j := i - 1
        // Pindahkan elemen yang lebih kecil dari key ke
posisi selanjutnya
```

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

// Fungsi untuk mencetak buku terfavorit (rating tertinggi)
func CetakTerfavorit(pustaka DaftarBuku) {
    if len(pustaka) > 0 {
        buku := pustaka[0]
        fmt.Printf("Buku Terfavorit:\nJudul: %s, Penulis:
%s, Penerbit: %s, Tahun: %d, Rating: %d\n",
            buku.Judul, buku.Penulis, buku.Penerbit,
buku.Tahun, buku.Rating)
    } else {
        fmt.Println("Tidak ada buku dalam daftar.")
    }
}

// Fungsi untuk mencetak 5 buku terbaru dengan rating
tertinggi
func Cetak5Terbaru(pustaka DaftarBuku) {
    fmt.Println("5 Buku dengan Rating Tertinggi:")
    for i := 0; i < 5 && i < len(pustaka); i++ {
        buku := pustaka[i]
        fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s,
Tahun: %d, Rating: %d\n",
            buku.Judul, buku.Penulis, buku.Penerbit,
buku.Tahun, buku.Rating)
    }
}

// Fungsi untuk mencari buku berdasarkan rating (binary
search)
func CariBuku(pustaka DaftarBuku, rating int) {
    left, right := 0, len(pustaka)-1
    for left <= right {
        mid := (left + right) / 2
        if pustaka[mid].Rating == rating {
            buku := pustaka[mid]
            fmt.Printf("Buku ditemukan:\nJudul: %s, Penulis:
%s, Penerbit: %s, Tahun: %d, Eksemplar: %d, Rating: %d\n",
```

```
                buku.Judul, buku.Penulis, buku.Penerbit,
buku.Tahun, buku.Eksemplar, buku.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() {
    // Input data buku
    pustaka := DaftarBuku{
        {1, "Buku A", "Penulis A", "Penerbit A", 10, 2020,
5},
        {2, "Buku B", "Penulis B", "Penerbit B", 5, 2018,
4},
        {3, "Buku C", "Penulis C", "Penerbit C", 3, 2019,
3},
        {4, "Buku D", "Penulis D", "Penerbit D", 2, 2021,
5},
        {5, "Buku E", "Penulis E", "Penerbit E", 7, 2022,
2},
        {6, "Buku F", "Penulis F", "Penerbit F", 1, 2023,
4},
    }

    // Mengurutkan buku berdasarkan rating
    pustaka = UrutBuku(pustaka)

    // Cetak buku terfavorit
    CetakTerfavorit(pustaka)

    // Cetak 5 buku terbaru
    Cetak5Terbaru(pustaka)

    // Cari buku berdasarkan rating
    fmt.Println("Mencari buku dengan rating 4:")
    CariBuku(pustaka, 4)

    fmt.Println("Mencari buku dengan rating 6:")
    CariBuku(pustaka, 6)
}
```

Screenshoot Output



Deskripsi Program

Program ini untuk mengelola data buku dalam sebuah perpustakaan. Program ini menggunakan struktur data Buku untuk menyimpan informasi tentang setiap buku, seperti judul, penulis, penerbit, tahun terbit, jumlah eksemplar, dan rating. Algoritma insertion sort digunakan untuk mengurutkan daftar buku berdasarkan rating secara descending (dari yang tertinggi ke terendah). Setelah diurutkan, program dapat dengan mudah mencari buku dengan rating tertentu menggunakan binary search, mencetak buku dengan rating tertinggi, atau mencetak 5 buku terbaru.