LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK

MODUL 11
PENGURUTAN DATA (SORTING)

Disusun Oleh:

NAMA : MARIA DWI A

NIM : 2311102228

KELAS : S1- 1F – 11 – 02

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

A. Pendahuluan

Pengurutan secara seleksi merupakan mencari nilai ekstrim dalam sekumpulan data, kemudian meletakkan pada posisi yang seharusnya. Data akan diurutkan secara memebesar (*ascending*), dan data dengan indeks kecil ada di “kiri” dan indeks besar ada di “kanan”. Langkah-langkah pencarian :

1) Cari nilai terkecil di dalam rentang data tersisa.
2) Pindahkan/tukar tempat dengan data yang berada pada posisi paling kiri pada rentang data yang tersisa.
3) Ulangi proses sampai tersisa hanya satu data.

Algortima ini dikenal juga dengan **Selection Sort**, dimana pada algoritma ini melibatkan data proses yaitu pencarian indeks dengan nilai ekstrim dan proses pertukaran dua nilai atau *swap*.

	Notasi Algoritma	Notasi dalam bahasa Go
1	i ← 1	i = 1
2	while i <= n-1 do	for i <= n-1 {
3	idx_min ← 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	}
11	t ← a[idx_min]	t = a[idx_min]
12	a[idx_min] ← a[i-1]	a[idx_min] = a[i-1]
13	a[i-1] ← t	a[i-1] = t
14	i ← i + 1	i = i + 1
15	endwhile	}

Contoh akgoritma *selection sort* untuk mengurutkan array bertipe data bilangan bulat secara membesar atau *ascending* :

```
..  ...
5   type arrInt [4321]int
..  ...
15  func selectionSort1(T *arrInt, n int){
16  /* I.S. terdefinisi array T yang berisi n bilangan bulat
17     F.S. array T terurut secara asceding atau membesar dengan SELECTION SORT */
18      var t, i, j, idx_min int
19
```

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

Sama seoerti jika elemen array yang akan diurutkan adalah bertipe data struct, maka cukup tambahakan 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 asceding 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
    }
}
```

B. Algoritma Insertion Sort

Pengurutan secara *insertion sort* adalah menyisipkan suatu nilai pada posisi yang seharusnya. Berbeda dengan pengurutan seleksi, yang mana pada pengurutan tidak dilakukan pencarian nilai ekstrim terlebih dahulu, cukup memilih suatu nilai tertentu kemudian mencari posisi secara *sequential search*. Contoh pengurutan data secara mengecil

(*descending*), dan data dengan indeks kecil berada di “kiri” dan indeks besar berada di “kanan”.

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

   Geser data yang sudah terurut tersebut (ke kanan), sehiingga ada suatu ruang yang kosong untuk memasukkan data yang sudah terurut ke dalam data yang sudah terurut dan tetap menjaga keterurutan.
2) Ulangi proses tersebut untuk setiap data yang belum terururt terurut terhadap rangkaian data yang sudah terurut.

Algoritma ini dikenal dengan nama ***Insertion Sort,*** dimana algoritma ini melibatkan dua proses yaitu pencarian sekuensial dan penyisipan.

	Notasi Algoritma	Notasi dalam bahasa Go
1	i ← 1	i = 1
2	while i <= n-1 do	for i <= n-1 {
3	j ← i	j = i
4	temp ← a[j]	temp = a[j]
5	while j > 0 and temp > a[j-1] do	for j > 0 && temp > a[j-1] {
6	a[j] ← a[j-1]	a[j] = a[j-1]
7	j ← j − 1	j = j − 1
8	endwhile	}
9	a[j] ← temp	a[j] = temp
10	i ← i + 1	i = i + 1
11	endwhile	}

Algoritma *insertion sort* untuk mengurutkan array bertipe data bilangan bulat secara mengecil atau *descending* adalah sebagai berikut :

```
..  ...
5   type arrInt [4321]int
..  ...
15  func insertionSort1(T *arrInt, n int){
16  /* I.S. terdefinisi array T yang berisi n bilangan bulat
17     F.S. array T terurut secara mengecil atau descending dengan INSERTION 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
28          i = i + 1
29      }
30  }
```

Seperti jika array yang akan diurutkan bertipe data struct, maka tambahkan field pada saat proses perbandingan dalam pencarian posisi, kemudian tipe data dari variabel ***temp*** sama dengan struct dari arraynya.

```
..  ...
5   type mahasiswa struct {
..      nama, nim, kelas, jurusan string
..      ipk float64
..  }
..  type arrMhs [2023]mahasiswa
..  ...
15  func insertionSort2(T * arrMhs, n int){
16  /* I.S. terdefinisi array T yang berisi n data mahasiswa
17     F.S. array T terurut secara mengecil atau descending berdasarkan nama dengan
18  menggunakan algoritma INSERTION SORT */
19      var temp i, j int
20      var temp mahasiswa
21      i = 1
22      for i <= n-1 {
23          j = i
24          temp = T[j]
25          for j > 0 && temp.nama > T[j-1].nama {
26              T[j] = T[j-1]
27              j = j - 1
28          }
29          T[j] = temp
30          i = i + 1
```

dilakukan pada array tersebut.Perhatikan potongan program dalam bahasa Go berikut ini :

```
5   type arrInt [2023]int
..  ...
15
16  func terkecil_1(tabInt arrInt, n int) int {
17  /* mengembalikan nilai terkecil yang terdapat di dalam tabInt yang berisi n
18  bilangan bulat */
19      var min int = tabInt[0]        // min berisi data pertama
20      var j int = 1                  // pencarian dimulai dari data berikutnya
21      for j < n {
22          if min > tabInt[j] {       // pengecekan apakah nilai minimum valid
23              min = tabInt[j]        // update nilai minimum dengan yang valid
24          }
25          j = j + 1
26      }
27      return min                     // returnkan nilai minimumnya
28  }
```

II. 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()
    }
}
```

Screenshoot Program

```
Masukkan jumlah daerah (n): 3
Masukkan jumlah rumah kerabat untuk daerah ke-1: 5 2 1 7 9 13
Masukkan nomor rumah kerabat untuk daerah ke-1: Hasil urutan rumah untuk daerah ke-1: 1 2 7 9 13
Masukkan jumlah rumah kerabat untuk daerah ke-2: 7
Masukkan nomor rumah kerabat untuk daerah ke-2: 6 189 15 27 39 75 133
Hasil urutan rumah untuk daerah ke-2: 6 15 27 39 75 133 189
Masukkan jumlah rumah kerabat untuk daerah ke-3: 4
Masukkan nomor rumah kerabat untuk daerah ke-3: 3 4 9 1
Hasil urutan rumah untuk daerah ke-3: 1 3 4 9
```

Deskripsi Program

Program ini digunakan untuk mengurutkan nomor rumah untuk beberapa daerah berdasarkan input pengguna. Program akan meminta untuk memasukkan jumlah daerah lalu untuk setiap daerah pengguna diminta untuk memasukkan jumlah rumah dan nomor rumah masing-masing. Program akan mengurutkan nomor rumah di setiap daerah dengan menggunakan metode SelectionSort. Setelah semua nomor rumah dari suatu daerah diurutkan, maka program akan menampilkan hasil dalam bentuk daftar nomor yang diurutkan dari kecil ke besar.

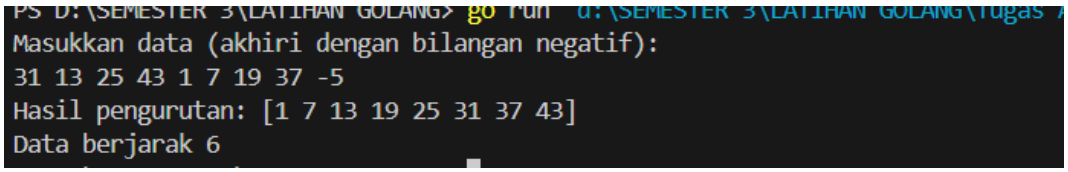
2. Source Code :

```
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")
    }
}
```

Screenshoot Program



Deskripsi Program

Program diatas merupakan implementasi dari algoritma pengurutan data insertion sort yang digunakan untuk menerima inputan bilangan bulat, lalu program akan mengurutkan data tersebut, dan memeriksa pakah elemen tersebut memiliki jarak yang tetap atau tidak. Program aka nmeminta pengguna untuk menginputkan bilangan dan proses akan berakhir ketika bilangan negatif diinputkan. Setelah bilangan diinputkan, selanjutnya program akan menggunakan insertion sort untuk mengurutkan data. Selanjutnya program akan memeriksa apakah data memilii jarak yang tetap atau tidak jika iya maka program akan mecetak besar jaraknya jika tidak program akan menampilkan output bahwa jarak tidak konsisten.

III. UNGUIDED Soal Modul 10

1. Source Code

```go
package main


import "fmt"

type arrayDaerah [1000]int

func selectionSort(array * arrayDaerah, n int) {
    var ganjil arrayDaerah
    var genap arrayDaerah
    var jumlahGanjil, jumlahGenap int

    for i:= 0; i < n; i++ {
        if array[i]%2 != 0{
            ganjil[jumlahGanjil] = array[i]
            jumlahGanjil++
        } else {
            genap[jumlahGenap] = array[i]
            jumlahGenap++
        }
    }

    for i:= 0; i < jumlahGanjil; i++ {
        minimal := i
        for j := i + 1; j < jumlahGanjil; j++{
            if ganjil[minimal] > ganjil[j] {
                minimal = j
            }
        }
        ganjil[i], ganjil[minimal] = ganjil[minimal], ganjil[i]
    }

    for i := 0; i < jumlahGenap; i++ {
        maksimal := i
        for j := i + 1; j < jumlahGenap; j++ {
            if genap[maksimal] < genap[j] {
                maksimal = j
            }
        }
        genap[i], genap[maksimal] = genap[maksimal], genap[i]
    }

    index := 0
    for i:= 0; i < jumlahGanjil; i++ {
        array[index] = ganjil[i]
        index++
    }

    for i := 0; i < jumlahGenap; i++ {
        array[index] = genap[i]
        index++
    }
}

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 jumlah int
        fmt.Printf("\nMasukkan jumlah kerabat untuk daerah ke-%d: ", i+1)
        fmt.Scan(&jumlah)


        if jumlah <= 0 || jumlah > 1000000 {
            fmt.Println("Jumlah rumah harus lebih besar dari 0 dan kurang dari
1000000!")
            return
        }
        var rumah arrayDaerah
        fmt.Printf("Masukkan nomor rumah kerabat daerah ke-%d: ", i+1)

        for j := 0; j < jumlah; j++ {
            fmt.Scan(&rumah[j])
        }
        selectionSort(&rumah, jumlah)

        fmt.Printf("Hasil urutan rumah untuk daerah ke-%d: ", i + 1)
        for k := 0; k < jumlah; k++ {
            fmt.Printf("%d ", rumah[k])
        }
        fmt.Println()
    }
}
```

Screenshoot Program

```
Masukkan jumlah daerah (n): 3

Masukkan jumlah kerabat untuk daerah ke-1: 6
Masukkan nomor rumah kerabat daerah ke-1: 5 2 1 7 9 13
Hasil urutan rumah untuk daerah ke-1: 1 5 7 9 13 2

Masukkan jumlah kerabat untuk daerah ke-2: 7
Masukkan nomor rumah kerabat daerah ke-2: 6 189 15 27 39 75 133
Hasil urutan rumah untuk daerah ke-2: 15 27 39 75 133 189 6

Masukkan jumlah kerabat untuk daerah ke-3: 4
Masukkan nomor rumah kerabat daerah ke-3: 3 4 9 1
Hasil urutan rumah untuk daerah ke-3: 1 3 9 4
```

Deskripsi Program

Program diatas merupakan implementasi dari Selection Sort yang digunakan untuk mengelompokkan dan mengurutkan nomor rumah berdasarkan kategori bilangan ganjil dan genap dalam beberapa daerah. Program akan memina pengguna untuk memasukkan jumlah daerah, lalu

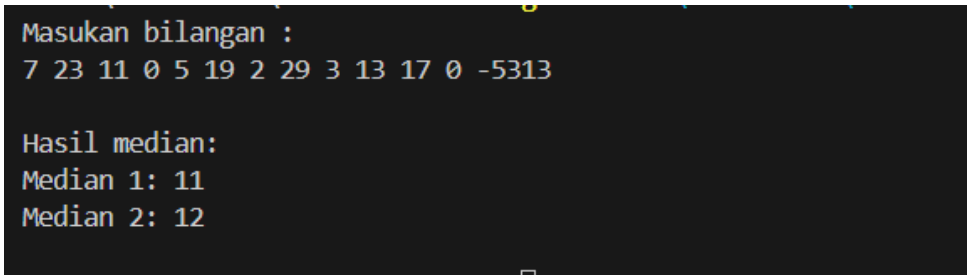
setiap daerah diisi dengan sejumlah nomor rumah. Setelah nomor dimasukkan maka nomor akan dibagi menjadi dua kelompok bilangan ganjil dan bilangan genap. Bilangan ganjil akan diurutkan dari yang terkecil ke besar dan bilangan genap akan diurutkan dari yang terbesar ke terkecil. Lalu program akan menampilkan hasil pengurutan setelah semua nomor dari masing masing daerah diinputkan.

2. Source code

```go
package main

import "fmt"


func hitungMedian(data []int, panjang int) float64{
    if panjang%2 == 1 {
        return float64(data[panjang/2])
    }
    return float64(data[panjang/2-1]+data[panjang/2]) / 2.0
}

func SelectionSort(data []int, panjang int) {
    for i:= 0; i < panjang - 1; i ++ {
        minimal := i
        for j := i; j < panjang; j++ {
            if data[j] < data[minimal] {
                minimal = j
            }
        }
        data[i], data[minimal] = data[minimal], data[i]
    }
}

func main() {
    var input int
    var data [1000000]int
    var semuaMedian[1000000]float64
    panjang := 0
    totalMedian := 0

    fmt.Println("Masukan bilangan : ")

    for {
        fmt.Scan(&input)

        if input == -5313 {
            break
        }

        if input == 0 {
            if panjang > 0 {
                SelectionSort(data[:], panjang)

                median := hitungMedian(data[:], panjang)
                semuaMedian[totalMedian] = median
                totalMedian++
            }
```

```
        } else {
            data[panjang] = input
            panjang++
        }
    }

    fmt.Println("\nHasil median: ")
    for i := 0; i < totalMedian; i++ {
        fmt.Printf("Median %d: %.0f\n", i+1, semuaMedian[i])
    }
    fmt.Println()
}
```

Screenshoot Program



Deskripsi Program

Program diatas adalah program yang digunakan untuk menghitung dan menampilkan median dari sejumlah data yang diinputkan pengguna. Pengguna bisa menginputkan data berupa bilangan bulat dan setiap kali diinputkan angka 0, program akan menghitung dan menyimpan nilai median berdasarkan data yang telah dimasukkan hingga angka tersebut. Setelah pengguna selesai menginputkan angka -5313 untuk mengakhiri masukan, lalu program akan menampilkan hasil median yang telah dihitung sebelumnya.

3. Source Code

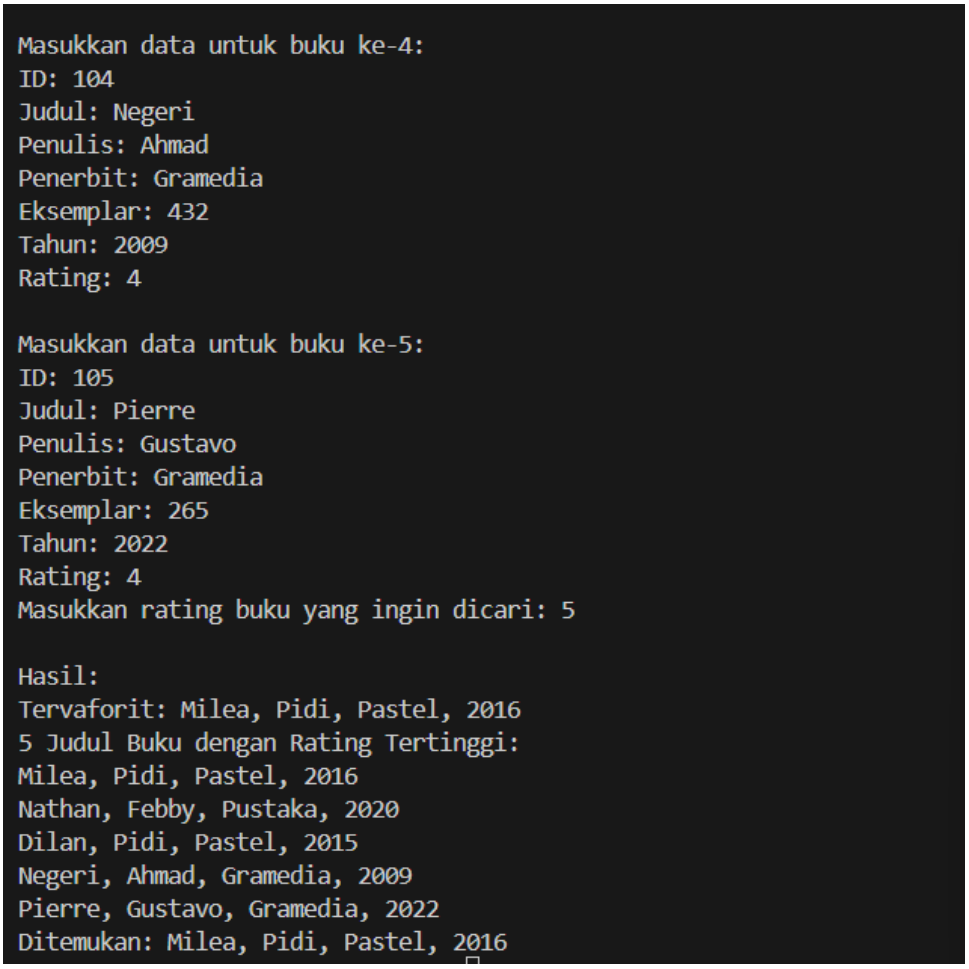
```
package main

import (
    "bufio"
    "fmt"
    "os"

)

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

const nMax int = 7919

type DaftarBuku [nMax]Buku

func cetakTerfavorit(pustaka DaftarBuku, n int) {
    if n == 0 {
        fmt.Println("Tidak ada buku di pustaka")
        return
    }

    ratingTertinggi := 0
    for i := 1; i < n; i++ {
        if pustaka[i].Rating > pustaka[ratingTertinggi].Rating {
            ratingTertinggi = i
        }
    }
    buku := pustaka[ratingTertinggi]
    fmt.Printf("Tervaforit: %s, %s, %s, %d\n", buku.Judul, buku.Penulis,
buku.Penerbit, buku.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) {
    batas := 5
    if n < 5 {
        batas = n
    }
    fmt.Println("5 Judul Buku dengan Rating Tertinggi:")
    for i := 0; i < batas; i++ {
        fmt.Printf("%s, %s, %s, %d\n", pustaka[i].Judul, pustaka[i].Penulis,
pustaka[i].Penerbit, pustaka[i].Tahun)
```

```
    }
}

func CariBuku(pustaka DaftarBuku, n int, rating int) {
    kiri := 0
    kanan := n - 1
    for kiri <= kanan {
        tengah := (kiri + kanan) / 2
        if pustaka[tengah].Rating == rating {
            buku := pustaka[tengah]
            fmt.Printf("Ditemukan: %s, %s, %s, %d\n", buku.Judul, buku.Penulis,
buku.Penerbit, buku.Tahun)
            return
        } else if pustaka[tengah].Rating < rating {
            kanan = tengah - 1
        } else {
            kiri = tengah + 1
        }
    }
    fmt.Println("Tidak ada buku dengan rating seperti itu.")
}

func main() {
    var pustaka DaftarBuku
    var n int
    scanner := bufio.NewScanner(os.Stdin)

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

    for i := 0; i < n; i++ {
        fmt.Printf("\nMasukkan data untuk buku ke-%d:\n", i+1)

        fmt.Print("ID: ")
        fmt.Scanln(&pustaka[i].ID)

        fmt.Print("Judul: ")
        if scanner.Scan() {
            pustaka[i].Judul = scanner.Text()
        }

        fmt.Print("Penulis: ")
        if scanner.Scan() {
            pustaka[i].Penulis = scanner.Text()
        }

        fmt.Print("Penerbit: ")
        if scanner.Scan() {
            pustaka[i].Penerbit = scanner.Text()
        }

        fmt.Print("Eksemplar: ")
        fmt.Scanln(&pustaka[i].Eksemplar)

        fmt.Print("Tahun: ")
        fmt.Scanln(&pustaka[i].Tahun)

        fmt.Print("Rating: ")
        fmt.Scanln(&pustaka[i].Rating)
    }

    var cariRating int
    fmt.Print("Masukkan rating buku yang ingin dicari: ")
    fmt.Scanln(&cariRating)

    fmt.Println("\nHasil:")
```

```
    cetakTerfavorit(pustaka, n)
    UrutBuku(&pustaka, n)
    Cetak5Terbaru(pustaka, n)
    CariBuku(pustaka, n, cariRating)
}
```

Screenshoot Program

```
Masukkan jumlah buku: 5

Masukkan data untuk buku ke-1:
ID: 101
Judul: Milea
Penulis: Pidi
Penerbit: Pastel
Eksemplar: 360
Tahun: 2016
Rating: 5

Masukkan data untuk buku ke-2:
ID: 102
Judul: Dilan
Penulis: Pidi
Penerbit: Pastel
Eksemplar: 400
Tahun: 2015
Rating: 4

Masukkan data untuk buku ke-3:
ID: 103
Judul: Nathan
Penulis: Febby
Penerbit: Pustaka
Eksemplar: 350
Tahun: 2020
Rating: 5

Masukkan data untuk buku ke-4:
ID: 104
```



Deskripsi Program

Program diatas merupakan program yang digunakan untuk mengelola data buku di dalam suatu perpustakaan. Inputan program terdiri dari bilangan bulat yang menyatakan banyak data buku yang ada di dalam perpustakaan. Lalu program akan meminta pengguna untuk menginputkan masing-masing dari data buku yang sesuai dengan atribut. Lalu program akan meminta pengguna untuk menginputkan rating buku yang akan dicari.

Lalu output program akan menampilkan buku dengan data terfavorit , baris kedua akan menampilkan judul dengan rating tertinggi, selanjutnya baris terakhir program akan menampilkan data buku yang dicari sesuai rating.