

LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2
MODUL XII & XIII
PENGURUTAN DATA



Oleh:

ANISSA FAUZIA ISYANTI

2311102219

S1IF-11-02

S1 TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024

I. DASAR TEORI

A. 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 *swap*.

	Notasi Algoritma	Notasi dalam bahasa Go
1	$i \leftarrow 1$	$i = 1$
2	while $i \leq n-1$ do	for $i \leq n-1$ {
3	$idx_min \leftarrow i - 1$	$idx_min = i - 1$
4	$j \leftarrow i$	$j = i$
5	while $j < n$ do	for $j < n$ {
6	if $a[idx_min] > a[j]$ then	if $a[idx_min] > a[j]$ {
7	$idx_min \leftarrow j$	$idx_min = j$
8	endif	}
9	$j \leftarrow j + 1$	$j = j + 1$
10	endwhile	}
11	$t \leftarrow a[idx_min]$	$t = a[idx_min]$
12	$a[idx_min] \leftarrow a[i-1]$	$a[idx_min] = a[i-1]$
13	$a[i-1] \leftarrow t$	$a[i-1] = t$
14	$i \leftarrow i + 1$	$i = i + 1$
15	endwhile	}

B. Algoritma Selection Sort

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

```
..    ...
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 ascending atau membesar dengan SELECTION SORT */
18      var t, i, j, idx_min int
19
```

```

20     i = 1
21     for i <= n-1 {
22         idx_min = i - 1
23         j = i
24         for j < n {
25             if T[idx_min] > T[j] {
26                 idx_min = j
27             }
28             j = j + 1
29         }
30         t = T[idx_min]
31         T[idx_min] = T[i-1]
32         T[i-1] = t
33         i = i + 1
34     }
35 }

```

Sama halnya apabila array yang akan diurutkan adalah tipe data struct, maka tambahkan fiels pada saat proses perbandingan nilai ekstrim, kemudian tipe data dari variabel t sama dengan struct dari arraynya.

```

..    ...
5    type mahasiswa struct {
..        nama, nim, kelas, jurusan string
..        ipk float64
..    }
..    type arrMhs [2023]mahasiswa
..    ...
15   func selectionSort2(T * arrMhs, n int){
16   /* I.S. terdefinisi array T yang berisi n data mahasiswa
17      F.S. array T terurut secara ascending atau membesar berdasarkan ipk dengan
18      menggunakan algoritma SELECTION SORT */
19      var i, j, idx_min int
20      var t mahasiswa
21      i = 1
22      for i <= n-1 {
23          idx_min = i - 1
24          j = i
25          for j < n {
26              if T[idx_min].ipk > T[j].ipk {
27                  idx_min = j
28              }
29              j = j + 1
30          }
31          t = T[idx_min]
32          T[idx_min] = T[i-1]
33          T[i-1] = t
34          i = i + 1
35      }
36  }

```

C. 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 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 erurut terhadap rangkaian data yang sudah terurut.

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

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

D. Algoritma Insertion Sort

Adapun algoritma *Insertion Sort* pada untuk mengurutkan array bertipe data bilangan bulat secara mengecil atau *descending* sebagai berikut ini!

```
..    ...
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 }
```

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.

```
..    ...
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
31   }
32 }
```

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++ {
        minIdx := i
        for j := i + 1; j < n; j++ {
            if arr[j] < arr[minIdx] { // Cari
elemen terkecil
                                minIdx = j
                            }
                        }
        arr[i], arr[minIdx] = arr[minIdx],
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()
    }
}

```

Screenshot Program

```

C:\Users\LENOVO\Documents\Alpro 2\Anissa Fauzia Isyanti_2311102219_Modul12>go run "c:\Users\LENOVO\Documents
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: 6 189 15 27 39 75 133
Masukkan nomor rumah kerabat untuk daerah ke-2: Hasil urutan rumah untuk daerah ke-2: 15 27 39 75 133 189
Masukkan jumlah rumah kerabat untuk daerah ke-3: 3 4 9 1
Masukkan nomor rumah kerabat untuk daerah ke-3: Hasil urutan rumah untuk daerah ke-3: 1 4 9

```

Program ini untuk mengurutkan nomor rumah menggunakan algoritma selection sort. Pengguna diminta memasukkan jumlah daerah, kemudian untuk setiap daerah, dimasukkan jumlah rumah dan daftar nomor rumahnya. Input jumlah daerah harus lebih dari 0 dan kurang dari 1000, serta jumlah rumah lebih dari 0 dan kurang dari 1.000.000. Setelah data dimasukkan, program mengurutkan nomor rumah menggunakan selection sort dan mencetak hasilnya dalam urutan yang terurut untuk setiap daerah.

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")
    }
}

```

Screenshot Program

```

C:\Users\LENOVO\Documents\Alpro 2\Anissa Fauzia Isyanti_2311102219_Modul12>
ided\guided2.go"
Masukkan data (akhiri dengan bilangan negatif):
31 13 25 43 1 7 19 37 -5
Hasil pengurutan: [1 7 13 19 25 31 37 43]
Data berjarak 6

```

```

C:\Users\LENOVO\Documents\Alpro 2\Anissa Fauzia Isyanti_2311102219_Modul12>
ided\tempCodeRunnerFile.go"
Masukkan data (akhiri dengan bilangan negatif):
1 4 5 6 3 13 -1
Hasil pengurutan: [1 3 4 5 6 13]
Data berjarak tidak tetap

```

Program ini merupakan program untuk mengurutkan bilangan menggunakan algoritma insertion sort. Di awal program, pengguna diminta memasukkan beberapa bilangan bulat hingga bilangan negatif dimasukkan sebagai penanda akhir input. Program kemudian mengurutkan bilangan dan memeriksa apakah data memiliki selisih elemen yang tetap. Jika jarak jarak antar elemen konsisten, program akan mencetak jarak selisih. Jika tidak, program akan mencetak pesan “Data berjarak tidak tetap”.

III. UNGUIDED

1. Source Code

```
package main

import "fmt"

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

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

func main() {
    var jmlDaerah int
    fmt.Println("Input")
    fmt.Scan(&jmlDaerah)

    if jmlDaerah <= 0 || jmlDaerah >= 1000 {
        fmt.Println("Jumlah daerah harus di antara 1 dan 999.")
        return
    }
    data := make([][]int, jmlDaerah)

    for i := 0; i < jmlDaerah; i++ {
        var jmlRumah int
        fmt.Scan(&jmlRumah)

        if jmlRumah <= 0 || jmlRumah >= 1000000 {
```

```

        fmt.Println("Jumlah rumah harus di
antara 1 dan 999999.")
        return
    }

    data[i] = make([]int, jmlRumah)
    for j := 0; j < jmlRumah; j++ {
        fmt.Scan(&data[i][j])
    }
}

fmt.Println("\nOutput:")
for _, daerah := range data {
    var ganjil []int
    var genap []int

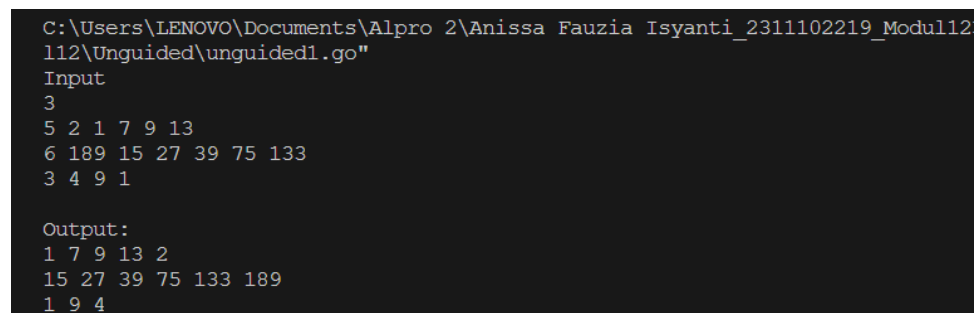
    for _, angka := range daerah {
        if angka%2 == 0 {
            genap = append(genap, angka)
        } else {
            ganjil = append(ganjil, angka)
        }
    }

    selectionSortAsc(ganjil)
    selectionSortDesc(genap)
    hasil := append(ganjil, genap...)

    for _, nilai := range hasil {
        fmt.Print(nilai, " ")
    }
    fmt.Println()
}
}

```

Screenshot Program



```

C:\Users\LENOVO\Documents\Alpro 2\Anissa Fauzia Isyanti_2311102219_Modul12
112\Unguided\unguided1.go"
Input
3
5 2 1 7 9 13
6 189 15 27 39 75 133
3 4 9 1

Output:
1 7 9 13 2
15 27 39 75 133 189
1 9 4

```

Program ini mengolah data rumah dalam beberapa daerah berdasarkan angka ganjil dan genap. Pengguna diminta memasukkan jumlah daerah (1–

999) dan jumlah rumah per daerah (1–999.999), diikuti dengan data angka rumah untuk masing-masing daerah. Program memisahkan angka menjadi dua kelompok: ganjil dan genap, lalu mengurutkan angka ganjil secara ascending dan angka genap secara descending menggunakan algoritma Selection Sort. Setelah itu, kedua kelompok angka digabungkan dengan urutan ganjil terlebih dahulu, diikuti genap, dan hasilnya ditampilkan untuk setiap daerah.

2. Source Code

```
package main

import "fmt"

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

func calculateMedian(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 input int
    var data []int

    for {
        fmt.Scan(&input)
        if input == -5313 {
            break
        }
        if input == 0 {
            if len(data) == 0 {
                continue
            }
        }
    }
}
```

```

        selectionSort(data)
        median := calculateMedian(data)
        fmt.Println(median)
    } else {
        data = append(data, input)
    }
}
}

```

Screenshot Program

```

C:\Users\LENOVO\Documents\Alpro 2\Anissa Fauzia Isyanti_2311102219_Modul12
112\Unguided\unguided2.go"
7 23 11 0 5 19 2 29 3 13 17 0 -5313
11
12

```

Program ini mengolah data angka dari input pengguna untuk menghitung nilai median. Pengguna dapat memasukkan serangkaian angka positif, di mana angka **0** digunakan untuk memicu perhitungan median dari angka-angka yang sudah dimasukkan sebelumnya, dan angka **-5313** digunakan untuk mengakhiri program. Ketika angka **0** dimasukkan, program akan mengurutkan angka yang sudah dimasukkan menggunakan algoritma *Selection Sort* dan menghitung median dari angka tersebut. Median dihitung sebagai angka tengah jika jumlah elemen ganjil atau rata-rata dari dua angka tengah jika jumlah elemen genap. Program ini terus menerima input hingga pengguna memasukkan **-5313**, dan akan mengabaikan input **0** jika belum ada data sebelumnya.

3. Source Code

```

package main

import "fmt"

const nMax = 7919

type Buku struct {
    id          int
    judul       string
    penulis     string
    penerbit    string
    eksemplar   int
    tahun       int
    rating      int
}

```

```

type DaftarBuku struct {
    pustaka []Buku
}

func DaftarkanBuku(pustaka *DaftarBuku, buku
Buku) {
    if len(pustaka.pustaka) < nMax {
        pustaka.pustaka =
append(pustaka.pustaka, buku)
    }
}

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

    terfavorit := pustaka.pustaka[0]
    for i := 1; i < n; i++ {
        if pustaka.pustaka[i].rating >
terfavorit.rating {
            terfavorit = pustaka.pustaka[i]
        }
    }

    return terfavorit
}

func UrutBuku(pustaka *DaftarBuku, n int) {
    for i := 1; i < n; i++ {
        key := pustaka.pustaka[i]
        j := i - 1
        for j >= 0 && pustaka.pustaka[j].rating
< key.rating {
            pustaka.pustaka[j+1] =
pustaka.pustaka[j]
            j--
        }
        pustaka.pustaka[j+1] = key
    }
}

func Cetak5Terbaru(pustaka DaftarBuku, n int) {
    if n == 0 {

```

```

        fmt.Println("Tidak ada buku dalam
pustaka.")
        return
    }

    fmt.Println("5 Buku Dengan Rating
Tertinggi:")
    for i := 0; i < 5 && i < n; i++ {
        buku := pustaka.pustaka[i]
        fmt.Printf("Judul : %s\nRating :
%d\n", buku.judul, buku.rating)
    }
}

func CariBuku(pustaka DaftarBuku, n, r int) {
    UrutBuku(&pustaka, n)

    low, high := 0, n-1
    for low <= high {
        mid := (low + high) / 2
        if pustaka.pustaka[mid].rating == r {
            buku := pustaka.pustaka[mid]
            fmt.Printf("Buku
ditemukan:\nJudul : %s\nPenulis :
%s\nPenerbit : %s\nTahun : %d\nEksemplar
: %d\nRating : %d\n",
                buku.judul, buku.penulis,
buku.penerbit, buku.tahun, buku.eksemplar,
buku.rating)
            return
        } else if pustaka.pustaka[mid].rating <
r {
            high = mid - 1
        } else {
            low = mid + 1
        }
    }
    fmt.Println("Tidak ada buku dengan rating
seperti itu.")
}

func main() {
    var pustaka DaftarBuku
    var n int

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

```



```

        for i := 0; i < n; i++ {
            var id, eksemplar, tahun, rating int
            var judul, penulis, penerbit string

            fmt.Printf("Masukkan data buku ke-
%d:\n", i+1)
            fmt.Print("ID          : ")
            fmt.Scanln(&id)
            fmt.Print("Judul       : ")
            fmt.Scanln(&judul)
            fmt.Print("Penulis    : ")
            fmt.Scanln(&penulis)
            fmt.Print("Penerbit   : ")
            fmt.Scanln(&penerbit)
            fmt.Print("Eksemplar  : ")
            fmt.Scanln(&eksemplar)
            fmt.Print("Tahun      : ")
            fmt.Scanln(&tahun)
            fmt.Print("Rating     : ")
            fmt.Scanln(&rating)

            buku := Buku{id, judul, penulis,
penerbit, eksemplar, tahun, rating}
            DaftarkanBuku(&pustaka, buku)
            fmt.Println()
        }

        terfavorit := CetakTerfavorit(pustaka, n)
        fmt.Println("Buku Terfavorit:")
        fmt.Printf("Judul          : %s\nPenulis        :
%s\nPenerbit      : %s\nTahun          : %d\nRating
: %d\n",
            terfavorit.judul, terfavorit.penulis,
            terfavorit.penerbit, terfavorit.tahun,
            terfavorit.rating)

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

        var targetRating int
        fmt.Print("\nMasukkan rating buku yang ingin
dicari: ")
        fmt.Scanln(&targetRating)
        CariBuku(pustaka, n, targetRating)
    }

```

Screenshot Program

```
C:\Users\LENOVO\Documents\Alpro 2\Anissa Fauzia Isyanti_2311102219_Modul12>go run "c:\Users\LENOVO\Documents\Alpro 2\Anissa Fauzia Isyanti_2311102219_Modul12\Unguided\unguided3.go"
Masukkan jumlah buku: 5
Masukkan data buku ke-1:
ID           : 1
Judul        : Seorang Lelaki yang Keluar dari Rumah
Penulis      : Leila S Chudori
Penerbit     : Kepustakaan Populer Gramedia
Eksemplar    : 1000
Tahun       : 2022
Rating      : 4

Masukkan data buku ke-2:
ID           : 2
Judul        : Tentang Kamu
Penulis      : Tere Liye
Penerbit     : Republika Penerbit
Eksemplar    : 100
Tahun       : 2016
Rating      : 1

Masukkan data buku ke-3:
ID           : 3
Judul        : Laut Bercerita
Penulis      : Leila S Chudori
Penerbit     : Kepustakaan Populer Gramedia
Eksemplar    : 5000
Tahun       : 2017
Rating      : 5

Masukkan data buku ke-4:
ID           : 4
Judul        : Pulan
Penulis      : Leila S Chudori
Penerbit     : Kepustakaan Populer Gramedia
Eksemplar    : 2500
Tahun       : 2012
Rating      : 3

Masukkan data buku ke-5:
ID           : 5
Judul        : Hujan
Penulis      : Tere Liye
Penerbit     : Republika Penerbit
Eksemplar    : 1500
Tahun       : 2016
Rating      : 2
```

```
5 Buku Dengan Rating Tertinggi:
Judul   : Laut Bercerita
Rating  : 5
Judul   : Seorang Lelaki yang Keluar dari Rumah
Rating  : 4
Judul   : Pulan
Rating  : 3
Judul   : Hujan
Rating  : 2
Judul   : Tentang Kamu
Rating  : 1

Masukkan rating buku yang ingin dicari: 5
Buku ditemukan:
Judul    : Laut Bercerita
Penulis  : Leila S Chudori
Penerbit : Kepustakaan Populer Gramedia
Tahun    : 2017
Eksemplar : 5000
Rating   : 5

C:\Users\LENOVO\Documents\Alpro 2\Anissa Fauzia Isyanti_2311102219_Modul12>
```

Program pengelolaan pustaka buku yang memungkinkan pengguna untuk memasukkan data buku, mengurutkan buku berdasarkan rating, dan melakukan pencarian buku berdasarkan rating tertentu. Program ini mendefinisikan dua tipe data utama, yaitu Buku yang berisi informasi terkait buku seperti ID, judul, penulis, penerbit, jumlah eksemplar, tahun terbit, dan rating; serta DaftarBuku yang merupakan koleksi dari buku-buku yang terdaftar. Pengguna dapat memasukkan sejumlah buku ke dalam pustaka, di mana data buku dapat berupa judul, penulis, penerbit, eksemplar, tahun terbit, dan rating. Program ini menyediakan berbagai fungsi: DaftarkanBuku untuk menambahkan buku ke pustaka, CetakTerfavorit untuk menampilkan buku dengan rating tertinggi, UrutBuku untuk mengurutkan buku berdasarkan rating tertinggi, Cetak5Terbaru untuk mencetak lima buku dengan rating tertinggi, dan CariBuku untuk mencari buku berdasarkan rating menggunakan pencarian biner. Program ini juga menggunakan bufio.Scanner untuk menangani input string yang lebih baik.