LAPORAN HASIL PRAKTIKUM ALGORITMA

DAN STRUKTUR DATA PRAKTIKUM 11

Oleh:
RANDA HERU KUSUMA

NIM. 2341760009

SIB-1F / 25

D-IV SISTEM INFORMASI BISNIS JURUSAN
TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG

Link github: https://github.com/randaheru/Praktikum11.git

12.2 Kegiatan Praktikum 1

12.2.1 Percobaan 1

Buat class di dalam paket tersebut dengan nama Node

```
public class Node {
```

. Di dalam class tersebut, deklarasikan atribut sesuai dengan diagram class di atas.

```
int data;
Node prev, next;
```

Selanjutnya tambahkan konstruktor default pada class Node sesuai diagram di atas.

```
Node(Node prev, int data, Node next) {
    this.prev = prev;
    this.data = data;
    this.next = next;
}
```

Buatlah sebuah class baru bernama DoubleLinkedLists pada package yang sama dengan nodeseperti gambar berikut:

```
public class DoubleLinkedLists {
```

Pada class DoubleLinkedLists tersebut, deklarasikan atribut sesuai dengan diagram class di atas.

```
Node head;
int size;
```

Selajuntnya, buat konstruktor pada class DoubleLinkedLists sesuai gambar berikut.

```
public DoubleLinkedLists() {
    head = null;
    size = 0;
}
```

Buat method isEmpty(). Method ini digunakan untuk memastikan kondisi linked list kosong.

```
public boolean isEmpty() {
    return head == null;
}
```

Kemudian, buat method addFirst(). Method ini akan menjalankan penambahan data di bagdepan linked list.

```
public void addFirst(int item) {
    if (isEmpty()) {
        head = new Node(prev:null, item, next:null);
    } else {
        Node newNode = new Node(prev:null, item, head);
        head.prev = newNode;
        head = newNode;
    }
    size++;
}
```

Selain itu pembuatan method addLast() akan menambahkan data pada bagian belakang linked list.

```
public void addLast(int item) {
    if (isEmpty()) {
        addFirst(item);
    } else {
        Node current = head;
        while (current.next != null) {
            current = current.next;
        }
        Node newNode = new Node(current, item, next:null);
        current.next = newNode;
        size++;
    }
}
```

Untuk menambahkan data pada posisi yang telah ditentukan dengan indeks, dapat dibuat dengan method add(int item, int index)

```
public void add(int item, int index) throws Exception {
    if (isEmpty()) {
        addFirst(item);
    } else if (index < 0 || index > size) {
        throw new Exception(message:"Nilai indeks di luar batas");
    } else {
        Node current = head;
        int i = 0;
        while (i < index) {
            current = current.next;
            i++;
        }

        if (current.prev == null) {
            Node newNode = new Node(prev:null, item, current);
            current.prev = newNode;
            head = newNode;
        } else {
            Node newNode = new Node(current.prev, item, current);
            current.prev.next = newNode;
            current.prev = newNode;
        }
        size++;
    }
}
```

Jumlah data yang ada di dalam linked lists akan diperbarui secara otomatis, sehingga dapat dibuat method size() untuk mendapatkan nilai dari size.

```
public int size() {
    return size;
}
```

Selanjutnya dibuat method clear() untuk menghapus semua isi linked lists, sehingga linked lists dalam kondisi kosong.

```
public void clear() {
    head = null;
    size = 0;
}
```

Untuk mencetak isi dari linked lists dibuat method print(). Method ini akan mencetak isi linked lists berapapun size-nya. Jika kosong akan dimunculkan suatu pemberitahuan bahwa linked lists dalam kondisi kosong.

```
public void print() {
    if (!isEmpty()) {
        Node tmp = head;
        while (tmp != null) {
            System.out.print(tmp.data + "\t");
            tmp = tmp.next;
        }
        System.out.println(x:"\nberhasil diisi");
    } else {
        System.out.println(x:"Linked Lists Kosong");
    }
}
```

Selanjutya dibuat class Main DoubleLinkedListsMain untuk mengeksekusi semua method yang ada pada class DoubleLinkedLists.

```
public class DoubleLinkedListsMain {
```

Pada main class pada langkah 16 di atas buatlah object dari class DoubleLinkedLists kemudian eksekusi potongan program berikut ini.

```
public static void main(String[] args) {
    DoubleLinkedLists dll = new DoubleLinkedLists();

    dll.print();
    System.out.println("Size : " + dll.size());
    System.out.println(x:"=====================================");

    dll.addFirst(item:3);
    dll.addLast(item:4);
    dll.addFirst(item:7);
    dll.print();
    System.out.println("Size : " + dll.size());
    System.out.println(x:"=====================================");

    try {
        dll.add(item:40, index:1);
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
    dll.print();
    System.out.println("Size : " + dll.size());
    System.out.println(x:"=====================================");

    try {
        dll.remove(index:1);
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
    dll.print();
    System.out.println("Size : " + dll.size());
    System.out.println(x:"=====================================");
```

12.2.2 Verifikasi Hasil Percobaan

```
Linked Lists Kosong
Size : 0
=====================================
7       3       4
berhasil diisi
Size : 3
=====================================
7       40      3       4
berhasil diisi
Size : 4
=====================================
Linked Lists Kosong
Size : 0
```

12.2.3 Pertanyaan Percobaan

1. Jelaskan perbedaan antara single linked list dengan double linked lists!

Pada single linked list, kita hanya dapat menavigasi maju melalui daftar, dimulai dari head dan berakhir di tail. Sedangkan Pada double linked list, kita dapat menavigasi maju dan mundur dalam daftar. Kita dapat dengan mudah melompat dari satu node ke node lainnya baik ke depan maupun ke belakang.

2. Perhatikan class Node, di dalamnya terdapat atribut next dan prev. Untuk apakah atribut tersebut?

Atribut next dan prev dalam class Node digunakan untuk menyimpan referensi ke node berikutnya dan node sebelumnya dalam double linked list. Dengan memiliki atribut next dan prev, setiap node dalam double linked list dapat terhubung dengan node sebelumnya dan node berikutnya, memungkinkan navigasi maju dan mundur dalam daftar.

3. Perhatikan konstruktor pada class DoubleLinkedLists. Apa kegunaan inisialisasi atribut head dan size seperti pada gambar berikut ini?

```
public DoubleLinkedLists() {
    head = null;
    size = 0;
}
```

inisialisasi atribut head ke null dan size ke 0 pada konstruktor adalah langkah penting untuk memastikan kestabilan dan konsistensi objek dari kelas DoubleLinkedLists.

4. Pada method addFirst(), kenapa dalam pembuatan object dari konstruktor class Node prev dianggap sama dengan null?

Node newNode = new Node(null, item, head);

Karena node pertama tidak memiliki node sebelumnya, maka kita mengatur nilai prev dari node baru ke null.

5. Perhatikan pada method addFirst(). Apakah arti statement head.prev = newNode ?

menunjukkan bahwa kita mengatur pointer prev dari node yang saat ini menjadi kepala (head) dari daftar ke node baru yang ditambahkan sebagai node pertama dalam daftar.

6. Perhatikan isi method addLast(), apa arti dari pembuatan object Node dengan mengisikan parameter prev dengan current, dan next dengan null?

Node newNode = new Node(current, item, null);

Digunakan untuk membuat node baru yang akan ditambahkan di akhir daftar (tail). Mari kita jelaskan arti dari setiap parameter yang diberikan ke konstruktor Node

7. Pada method add(), terdapat potongan kode program sebagai berikut:

```
while (i < index) {
    current = current.next;
    i++;
}
if (current.prev == null) {
    Node newNode = new Node(null, item, current);
    current.prev = newNode;
    head = newNode;
} else {
    Node newNode = new Node(current.prev, item, current);
    newNode.prev = current.prev;
    newNode.next = current;
    current.prev.next = newNode;
    current.prev = newNode;
}
```

jelaskan maksud dari bagian yang ditandai dengan kotak kuning.

Pernyataan if (current.prev == null) menunjukkan bahwa kita sedang memeriksa apakah current, yaitu node terakhir dalam daftar (tail) sebelumnya, memiliki node sebelumnya atau tidak. Jika current.prev == null, ini berarti bahwa current adalah node pertama dalam daftar atau daftar saat ini kosong.

12.3 Kegiatan Praktikum 2

12.3.1 Tahapan Percobaan

Buatlah method removeFirst() di dalam class DoubleLinkedLists.

```
public void removeFirst() throws Exception {
    if (isEmpty()) {
        throw new Exception(message:"Linked List masih kosong, tidak dapat dihapus!");
    } else if (size == 1) {
        head = null;
        size--;
    } else {
        head = head.next;
        head.prev = null;
        size--;
    }
}
```

Tambahkan method removeLast() di dalam class DoubleLinkedLists.

```
public void removeLast() throws Exception {
    if (isEmpty()) {
        throw new Exception(message:"Linked List masih kosong, tidak dapat dihapus!");
    } else if (head.next == null) {
        head = null;
        size--;
    } else {
        Node current = head;
        while (current.next.next != null) {
            current = current.next;
        }
        current.next = null;
        size--;
    }
}
```

inkedLists dan amati hasilnya.

```
public void remove(int index) throws Exception {
    if (isEmpty() || index >= size || index < 0) {
        throw new Exception(message:"Nilai indeks di luar batas");
    } else if (index == 0) {
        removeFirst();
    } else if (index == size - 1) {
        removeLast();
    } else {
        Node current = head;
        int i = 0;
        while (i < index) {
            current = current.next;
            i++;
        }
        current.prev.next = current.next;
        current.next.prev = current.prev;
        size--;
    }
}
```

Untuk mengeksekusi method yang baru saja dibuat, tambahkan potongan kode program berikut pada main class.

```
dll.addLast(item:50);
dll.addLast(item:40);
dll.addLast(item:10);
dll.addLast(item:20);
dll.print();
System.out.println("Size : " + dll.size());
System.out.println(x:"=====================================");

try {
    dll.removeFirst();
} catch (Exception e) {
    System.out.println(e.getMessage());
}
dll.print();
System.out.println("Size : " + dll.size());
System.out.println(x:"=====================================");

try {
    dll.removeLast();
} catch (Exception e) {
    System.out.println(e.getMessage());
}
dll.print();
System.out.println("Size : " + dll.size());
System.out.println(x:"=====================================");

try {
    dll.remove(index:1);
} catch (Exception e) {
    System.out.println(e.getMessage());
}
dll.print();
```

12.3.2 Verifikasi Hasil Percobaan

```
Linked Lists Kosong
Size : 0
=====================================
7       3       4
berhasil diisi
Size : 3
=====================================
7       40      3       4
berhasil diisi
Size : 4
=====================================
7       3       4
berhasil diisi
Size : 3
=====================================
3       4
berhasil diisi
Size : 2
=====================================
3
berhasil diisi
Size : 1
```

12.3.3 Pertanyaan Percobaan

1. Apakah maksud statement berikut pada method removeFirst()?

head = head.next;

head.prev = null;

Ini dilakukan karena setelah head saat ini dihapus, tidak ada node sebelumnya dalam daftar yang akan menjadi head baru, sehingga pointer prev dari node yang baru diatur ke null untuk menandakan bahwa sekarang node tersebut adalah head dari daftar dan tidak memiliki node sebelumnya.

2. Bagaimana cara mendeteksi posisi data ada pada bagian akhir pada method removeLast()?

perlu memeriksa apakah node yang sedang diproses adalah node terakhir dalam daftar. Jika ya, maka kita tahu bahwa kita berada pada bagian akhir.

3. Jelaskan alasan potongan kode program di bawah ini tidak cocok untuk perintah remove!

```
Node tmp = head.next;

head.next=tmp.next;
tmp.next.prev=head;
```

Potongan kode tersebut mengabaikan langkah-langkah yang diperlukan untuk membebaskan memori dari node yang dihapus. Menghapus node tanpa membebaskan memori dapat menyebabkan kebocoran memori atau penghapusan data yang tidak diinginkan.

4. Jelaskan fungsi kode program berikut ini pada fungsi remove!

```
current.prev.next = current.next;
current.next.prev = current.prev;
```

fungsi remove yang bertujuan untuk menghapus node dari daftar yang terhubung dalam struktur data double linked list

12.4 Kegiatan Praktikum 3

12.4.1 Tahapan Percobaan

Buatlah method getFirst() di dalam class DoubleLinkedLists untuk mendapatkan data padaawal linked lists.

```
public int getFirst() throws Exception {
    if (isEmpty()) {
        throw new Exception(message:"Linked List kosong");
    }
    return head.data;
}
```

Selanjutnya, buatlah method getLast() untuk mendapat data pada akhir linked lists.

```
public int getLast() throws Exception {
    if (isEmpty()) {
        throw new Exception(message:"Linked List kosong");
    }

    Node tmp = head;
    while (tmp.next != null) {
        tmp = tmp.next;
    }

    return tmp.data;
}
```

Method get(int index) dibuat untuk mendapatkan data pada indeks tertentu

```
public int get(int index) throws Exception {
    if (isEmpty() || index >= size || index < 0) {
        throw new Exception(message:"Nilai indeks di luar batas.");
    }

    Node tmp = head;
    for (int i = 0; i < index; i++) {
        tmp = tmp.next;
    }

    return tmp.data;
}
```

Pada main class tambahkan potongan program berikut dan amati hasilnya!

```
        dll.print();
        System.out.println("Size: " + dll.size());
        System.out.println(x:"================================");

        dll.addFirst(item:3);
        dll.addLast(item:4);
        dll.addFirst(item:7);
        dll.print();
        System.out.println("Size: " + dll.size());
        System.out.println(x:"================================");

        try {
            dll.add(item:40, index:1);
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
        dll.print();
        System.out.println("Size: " + dll.size());
        System.out.println(x:"================================");

        try {
            System.out.println("Data awal pada Linked Lists adalah: " + dll.getFirst());
            System.out.println("Data akhir pada Linked Lists adalah: " + dll.getLast());
            System.out.println("Data indeks ke-1 pada Linked Lists adalah: " + dll.get(index:1));
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

12.4.3 Pertanyaan Percobaan

1. Jelaskan method size() pada class DoubleLinkedLists!

Metode size() pada kelas DoubleLinkedLists bertujuan untuk mengembalikan jumlah node yang ada dalam linked list saat ini. Ini adalah metode yang sederhana tetapi penting karena memberikan informasi tentang seberapa besar linked list

2. Jelaskan cara mengatur indeks pada double linked lists supaya dapat dimulai dari indeks ke-1!

Memodifikasi cara mengakses dan memanipulasi indeks saat menambah, menghapus, atau mengakses elemen dalam double linked list,

3. Jelaskan perbedaan karakteristik fungsi Add pada Double Linked Lists dan Single Linked Lists! Pada single linked list, setiap node hanya memiliki satu tautan yang menunjuk ke node berikutnya.

Pada double linked list, ketika menambahkan elemen baru, setiap node memiliki dua tautan, yaitu tautan ke node sebelumnya (prev) dan tautan ke node berikutnya (next). Sedangkan

4. Jelaskan perbedaan logika dari kedua kode program di bawah ini!

A

```
public boolean isEmpty(){
    if(size ==0){
        return true;
    } else{
        return false;
    }
}
```

B

```
public boolean isEmpty(){
    return head == null;
}
```

cara kondisi diungkapkan. Kode pertama menggunakan pendekatan yang sedikit lebih eksplisit dengan menggunakan if-else, sementara kode kedua menggunakan pendekatan yang lebih langsung dengan menggunakan ekspresi boolean tunggal

12.4.2 Verifikasi Hasil Percobaan

```
Linked Lists Kosong
Size : 0
==================================
50      40      10      20
berhasil diisi
Size : 4
==================================
40      10      20
berhasil diisi
Size : 3
==================================
40      10
berhasil diisi
Size : 2
==================================
40
berhasil diisi
Size : 1
==================================
40
berhasil diisi
Size: 1
================================
7       3       40      4
berhasil diisi
Size: 4
================================
7       40      3       40      4
berhasil diisi
Size: 5
================================
Data awal pada Linked Lists adalah: 7
Data akhir pada Linked Lists adalah: 4
Data indeks ke-1 pada Linked Lists adalah: 40
```

12.5 Tugas Praktikum

1. Buat program antrian vaksinasi menggunakan queue berbasis double linked list sesuai ilustrasi dan menu di bawah ini! (counter jumlah antrian tersisa di menu cetak(3) dan data orang yang telah divaksinasi di menu Hapus Data(2) harus ada)

Buat class Node

```java
public class Node {
    int nomorAntrian;
    String namaPenerima;
    Node prev, next;

    public Node(Node prev, int nomorAntrian, String namaPenerima, Node next) {
        this.prev = prev;
        this.nomorAntrian = nomorAntrian;
        this.namaPenerima = namaPenerima;
        this.next = next;
    }
}
```

Buat class Queue

```java
public class Queue {
    Node head, tail;
    int size;

    public Queue() {
        head = tail = null;
        size = 0;
    }

    public boolean isEmpty() {
        return head == null;
    }

    public void enqueue(int nomorAntrian, String namaPenerima) {
        Node newNode = new Node(tail, nomorAntrian, namaPenerima, next:null);
        if (isEmpty()) {
            head = tail = newNode;
        } else {
            tail.next = newNode;
            tail = newNode;
        }
        size++;
        System.out.println(x:"Data penerima vaksin berhasil ditambahkan.");
    }

```

Lalu buat class mainnya

```java
import java.util.Scanner;

public class VaccinationQueueMain {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Queue queue = new Queue();
        int choice;

        do {
            System.out.println(x:"\nPENGANTRI VAKSIN EXTRAVAGANZA\n");
            System.out.println(x:"1. Tambah Data Penerima Vaksin");
            System.out.println(x:"2. Hapus Data Pengantri Vaksin");
            System.out.println(x:"3. Daftar Penerima Vaksin");
            System.out.println(x:"4. Keluar");
            System.out.print(s:"\nPilih menu: ");
            choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    System.out.println(x:"\nMasukkan Data Penerima Vaksin");
                    System.out.print(s:"Nomor Antrian: ");
                    int nomorAntrian = scanner.nextInt();
                    scanner.nextLine(); // consume newline
                    System.out.print(s:"Nama Penerima: ");
                    String namaPenerima = scanner.nextLine();
                    queue.enqueue(nomorAntrian, namaPenerima);
                    break;
                case 2:
```

Hasil Output:

Menu awal dan penambahan data

```
PENGANTRI VAKSIN EXTRAVAGANZA

1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar

Pilih menu: 1

Masukkan Data Penerima Vaksin
Nomor Antrian: 123
Nama Penerima: Joko
Data penerima vaksin berhasil ditambahkan.
```

Cetak data

```
PENGANTRI VAKSIN EXTRAVAGANZA

1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar

Pilih menu: 3
Daftar Pengantri Vaksin:
No.     Nama
123     Joko
124     Mely
135     Johan
146     Rossi
Sisa Antrian: 4
```

Hapus data

```
PENGANTRI VAKSIN EXTRAVAGANZA

1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar

Pilih menu: 2
Data penerima vaksin dengan nomor antrian 123 dan nama Joko telah divaksinasi.

PENGANTRI VAKSIN EXTRAVAGANZA

1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar

Pilih menu:
```

2. Buatlah program daftar film yang terdiri dari id, judul dan rating menggunakan double linked lists, bentuk program memiliki fitur pencarian melalui ID Film dan pengurutan Rating secara descending. Class Film wajib diimplementasikan dalam soal ini.

Class Film

```java
public class Film {
    int id;
    String title;
    double rating;

    public Film(int id, String title, double rating) {
        this.id = id;
        this.title = title;
        this.rating = rating;
    }

    @Override
    public String toString() {
        return "ID: " + id + "\nJudul Film: " + title + "\nipk: " + rating;
    }
}
```

Class Node

```java
public class Node2 {
    Film data;
    Node2 prev, next;

    public Node2(Node2 prev, Film data, Node2 next) {
        this.prev = prev;
        this.data = data;
        this.next = next;
    }
}
```

```java
public class DoubleLinkedList {
    Node2 head, tail;
    int size;

    public DoubleLinkedList() {
        head = tail = null;
        size = 0;
    }

    public boolean isEmpty() {
        return head == null;
    }

    public void addFirst(Film data) {
        Node2 newNode = new Node2(prev:null, data, head);
        if (isEmpty()) {
            head = tail = newNode;
        } else {
            head.prev = newNode;
            head = newNode;
        }
        size++;
    }

    public void addLast(Film data) {
        if (isEmpty()) {
            addFirst(data);
        } else {
```

Class Main

```java
import java.util.Scanner;

public class Main {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        DoubleLinkedList list = new DoubleLinkedList();
        int choice;

        do {
            System.out.println(x:"\n==================");
            System.out.println(x:"DATA FILM LAYAR LEBAR");
            System.out.println(x:"==================");
            System.out.println(x:"1. Tambah Data Awal");
            System.out.println(x:"2. Tambah Data Akhir");
            System.out.println(x:"3. Tambah Data Index Tertentu");
            System.out.println(x:"4. Hapus Data Pertama");
            System.out.println(x:"5. Hapus Data Terakhir");
            System.out.println(x:"6. Hapus Data Tertentu");
            System.out.println(x:"7. Cetak");
            System.out.println(x:"8. Cari ID Film");
            System.out.println(x:"9. Urut Data Rating Film-DESC");
            System.out.println(x:"10. Keluar");
            System.out.println(x:"==================");
            System.out.print(s:"Pilih menu: ");
            choice = scanner.nextInt();

            switch (choice) {
```

Menu awal dan penambahan data

```
==================
DATA FILM LAYAR LEBAR
==================
1. Tambah Data Awal
2. Tambah Data Akhir
3. Tambah Data Index Tertentu
4. Hapus Data Pertama
5. Hapus Data Terakhir
6. Hapus Data Tertentu
7. Cetak
8. Cari ID Film
9. Urut Data Rating Film-DESC
10. Keluar
==================
Pilih menu: 1
ID Film: 1222
Judul Film: Spider-Man: No Way Home
Rating Film: 8.7
```

```
==================
DATA FILM LAYAR LEBAR
==================
1. Tambah Data Awal
2. Tambah Data Akhir
3. Tambah Data Index Tertentu
4. Hapus Data Pertama
5. Hapus Data Terakhir
6. Hapus Data Tertentu
7. Cetak
8. Cari ID Film
9. Urut Data Rating Film-DESC
10. Keluar
==================
Pilih menu: 2
ID Film: 1346
Judul Film: Uncharted
Rating Film: 6.7
```

```
==================
DATA FILM LAYAR LEBAR
==================
1. Tambah Data Awal
2. Tambah Data Akhir
3. Tambah Data Index Tertentu
4. Hapus Data Pertama
5. Hapus Data Terakhir
6. Hapus Data Tertentu
7. Cetak
8. Cari ID Film
9. Urut Data Rating Film-DESC
10. Keluar
==================
Pilih menu: 3
ID Film: 1234
Judul Film: Death in the Nile
Rating Film: 6.6
Urutan ke-: 3
Nilai indeks di luar batas
```

Cetak data

```
==================
DATA FILM LAYAR LEBAR
==================
1. Tambah Data Awal
2. Tambah Data Akhir
3. Tambah Data Index Tertentu
4. Hapus Data Pertama
5. Hapus Data Terakhir
6. Hapus Data Tertentu
7. Cetak
8. Cari ID Film
9. Urut Data Rating Film-DESC
10. Keluar
==================
Pilih menu: 7
ID: 1222
Judul Film: Spider-Man: No Way Home
ipk: 8.7
ID: 1346
Judul Film: Uncharted
ipk: 6.7
```

Pencarian data

```
==================
DATA FILM LAYAR LEBAR
==================
1. Tambah Data Awal
2. Tambah Data Akhir
3. Tambah Data Index Tertentu
4. Hapus Data Pertama
5. Hapus Data Terakhir
6. Hapus Data Tertentu
7. Cetak
8. Cari ID Film
9. Urut Data Rating Film-DESC
10. Keluar
==================
Pilih menu: 8
Masukkan ID Film yang dicari: 1222
Data Id Film: 1222 berada di node ke- 1
IDENTITAS:
ID: 1222
Judul Film: Spider-Man: No Way Home
ipk: 8.7
```