Laporan Hasil Pratikum

Algoritma Dan Struktur Data

Jobsheet 12

Nama : Zacky Rio Orlando

NIM : 244107020086

Kelas : 1E

Program Studi D-IV Teknik Informatika

Jurusan Teknologi Informasi

Praktikum

2025

12.2 Kegiatan Praktikum 1
12.2.1 Percobaan 1

Class Mahasiswa27

```
public class Mahasiswa27 {
    public String nim;
    public String nama;
    public String kelas;
    public double ipk;

    public Mahasiswa27(String nim, String nama, String kelas, double ipk) {
        this.nim = nim;
        this.nama = nama;
        this.kelas = kelas;
        this.ipk = ipk;
    }

    public void tampil() {
        System.out.println("NIM: " + nim + ", Nama: " + nama + ", Kelas: "
+ kelas + ", IPK: " + ipk);
    }
}
```

Class Node27

```
public class Node27 {
    Mahasiswa27 data;
    Node27 prev;
    Node27 next;

    public Node27(Mahasiswa27 data) {
        this.data = data;
        this.prev = null;
        this.next = null;
    }
}
```

Class DoubleLinkedList27

```
public class DoubleLinkedList27 {
    Node27 head;
    Node27 tail;

    public DoubleLinkedList27() {
        head = null;
        tail = null;
    }

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

    public void addFirst(Mahasiswa27 data) {
        Node27 newNode27 = new Node27(data);
        if (isEmpty()) {
            head = tail = newNode27;
        } else {
            newNode27.next = head;
            head.prev = newNode27;
            head = newNode27;
        }
    }

    public void addLast(Mahasiswa27 data) {
        Node27 newNode27 = new Node27(data);
        if (isEmpty()) {
            head = tail = newNode27;
        } else {
            tail.next = newNode27;
            newNode27.prev = tail;
            tail = newNode27;
        }
    }

    public void inserAfter(String keyNim, Mahasiswa27 data) {
        Node27 current = head;

        while (current != null && !current.data.nim.equals(keyNim)) {
            current = current.next;
        }

        if (current == null) {
            System.out.println("Node dengan NIM " + keyNim + " tidak
ditemukan.");
            return;
        }

        Node27 newNode27 = new Node27(data);
```

```
        if (current == tail) {
            current.next = newNode27;
            newNode27.prev = current;
            tail = newNode27;
        } else {
            newNode27.next = current.next;
            newNode27.prev = current;
            current.next.prev = newNode27;
            current.next = newNode27;
        }

        System.out.println("Node berhasil disisipkan setelah NIM " + keyNim);
    }

    public void print() {
        Node27 current = head;
        while (current != null) {
            current.data.tampil();
            current = current.next;
        }
    }

    public void removeFirst() {
        if (isEmpty()) {
            System.out.println("List kosong, tidak bisa dihapus.");
            return;
        }
        if (head == tail) {
            head = tail = null;
        } else {
            head = head.next;
            head.prev = null;
        }
    }

    public void removeLast() {
        if (isEmpty()) {
            System.out.println("List Kosong, tidak bisa dihapus.");
            return;
        }
        if (head == tail) {
            head = tail = null;
        } else {
            tail = tail.prev;
            tail.next = null;
        }
    }

    public Node27 search(String nim) {
        Node27 current = head;
        while (current != null) {
            if (current.data.nim.equals(nim)) {
                return current;
            }
            current = current.next;
        }
        return null;
    }
}
```

Class DLLMain

```
import java.util.Scanner;
public class DLLMain {
    public static Mahasiswa27 inputMahasiswa(Scanner scan) {
        System.out.print("Masukkan NIM: ");
        String nim = scan.nextLine();
        System.out.print("Masukkan Nama: ");
        String nama = scan.nextLine();
        System.out.print("Masukkan Kelas: ");
        String kelas = scan.nextLine();
        System.out.print("Masukkan IPK: ");
        double ipk = scan.nextDouble();
        scan.nextLine();
        return new Mahasiswa27(nim, nama, kelas, ipk);
    }
    public static void main(String[] args) {
        DoubleLinkedList27 list = new DoubleLinkedList27();
        Scanner scan = new Scanner(System.in);
        int pilihan;

        do {
            System.out.println("\nMenu Double Linked List Mahasiswa");
            System.out.println("1. Tambah di awal");
            System.out.println("2. Tambah di akhir");
            System.out.println("3. Hapus di awal");
            System.out.println("4. Hapus di akhir");
            System.out.println("5. Tampilkan data");
            System.out.println("7. Cari Mahasiswa berdasarkan NIM");
            System.out.println("0. Keluar");
            System.out.print("Pilih menu: ");
            pilihan = scan.nextInt();
            scan.nextLine();

            switch (pilihan) {
                case 1 -> {
                Mahasiswa27 mhs = inputMahasiswa(scan);
                list.addFirst(mhs);
                }
                case 2 -> {
                    Mahasiswa27 mhs = inputMahasiswa(scan);
                    list.addLast(mhs);
                }
                case 3 -> list.removeFirst();
                case 4 -> list.removeLast();
                case 5 -> list.print();
                case 6 -> {
                    System.out.println("Masukkan NIM yang dicari: ");
                    String nim = scan.nextLine();
                    Node27 found = list.search(nim);
                    if (found != null) {
                        System.out.println("Data ditemukan:");
                        found.data.tampil();
                    } else {
                        System.out.println("Data tidak ditemukan.");
                    }
                }
                case 0 -> System.out.println("Keluar dari program.");
                default -> System.out.println("Pilihan tidak valid!");
            }
        } while (pilihan != 0);
        scan.close();
    }
}
```

Hasil Output dari kode program diatas

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
7. Cari Mahasiswa berdasarkan NIM
0. Keluar
Pilih menu: 1
Masukkan NIM: 20304050
Masukkan Nama: Hermione
Masukkan Kelas: Gryffindor
Masukkan IPK: 4,0

Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
7. Cari Mahasiswa berdasarkan NIM
0. Keluar
Pilih menu: 5
NIM: 20304050, Nama: Hermione, Kelas: Gryffindor, IPK: 4.0
```

12.2.3 Pertanyaan Percobaan

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

Jawaban :

Single Linked List adalah susunan data di mana setiap bagian (node) hanya punya satu penunjuk, yaitu next, yang menunjukkan ke data berikutnya. Karena itu, kita cuma bisa jalan dari awal ke akhir secara searah. Sedangkan Double Linked List lebih lengkap, karena tiap bagian punya dua penunjuk, yaitu next dan prev, jadi kita bisa maju dan mundur. Ini bikin proses nambah atau hapus data di tengah jadi lebih gampang dan cepat. Tapi, Double Linked List butuh lebih banyak memori karena menyimpan dua penunjuk.

2. Perhatikan class Node01, di dalamnya terdapat atribut next dan prev. Untuk apakah atribut tersebut?

Jawaban :

Untuk setiap data bisa terhubung dengan data lain di depan dan di belakangnya. next itu menunjukkan data berikutnya, sedangkan prev menunjukkan data sebelumnya. Karena ada dua penunjuk, kita bisa jalan maju atau mundur di daftar data, jadi lebih mudah buat mencari, menambah, atau juga bisa menghapus data di tengah daftar.

3. Perhatikan konstruktor pada class DoubleLinkedLists. Apa kegunaan dari konstruktor tersebut?

```
public DoubleLinkedList01() {
    head = null;
    tail = null;
}
```

Jawaban :

Digunakan untuk membuat sebuah objek Double Linked List yang baru dengan kondisi awal kosong, yaitu dengan mengatur nilai head dan tail menjadi null. Jadi, saat pertama kali list dibuat, belum ada data sama sekali, jadi kedua penunjuk tersebut belum menunjuk ke node manapun.

4. Pada method **addFirst()**, apa maksud dari kode berikut?

```
if (isEmpty()) {
    head = tail = newNode;
```

Jawaban :

Yaitu, jika daftar (list) masih kosong (belum ada node sama sekali), maka node baru (newNode27) yang dibuat langsung menjadi node pertama sekaligus node terakhir di list. Jadi, baik penunjuk head (awal) maupun tail (akhir) sama-sama menunjuk ke node baru itu karena hanya ada satu node di list.

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

Jawaban :

Untuk menghubungkan node yang sekarang menjadi head dengan node baru (newNode) yang akan ditambahkan di depannya. Dengan kata lain, node baru tersebut dijadikan sebagai node sebelumnya (prev) dari node yang lama, supaya hubungan dua arah dalam Double Linked List tetap terjaga.

6. Modifikasi code pada fungsi **print()** agar dapat menampilkan warning/ pesan bahwa linked lists masih dalam kondisi.

Hasil Modifikasi

```
    public void print() {
        if (isEmpty()) {
            System.out.println("Linked List masih kosong, tidak ada data
yang ditampilkan.");
        } else {
            Node27 current = head;
            while (current != null) {
                current.data.tampil();
                current = current.next;
            }
        }
    }
```

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
7. Cari Mahasiswa berdasarkan NIM
0. Keluar
Pilih menu: 5
Linked List masih kosong, tidak ada data yang ditampilkan.
```

7. Pada insertAfter(), apa maksud dari kode berikut ?

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

Jawaban :
Untuk mengubah penunjuk prev dari node yang berada setelah current agar menunjuk ke node baru (newNode) yang baru disisipkan setelah current. Jadi, ini untuk memastikan node setelahnya tahu bahwa node sebelumnya sekarang adalah newNode, supaya hubungan dua arah di Double Linked List tetap benar dan rapi.

8. Modifikasi menu pilihan dan switch-case agar fungsi insertAfter() masuk ke dalam menu pilihan dan dapat berjalan dengan baik.

Jawaban :

Hasil modifikasinya

```
System.out.println("7. Sisipkan setelah NIM tertentu");
case 7 -> {
                    System.out.print("Masukkan NIM setelah node yang ingin
disisipkan: ");
                    String keyNim = scan.nextLine();
                    System.out.println("Masukkan data mahasiswa yang akan
disisipkan:");
                    Mahasiswa27 mhs = inputMahasiswa(scan);
                    list.inserAfter(keyNim, mhs);
                }
```

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
0. Keluar
Pilih menu: 1
Masukkan NIM: 20304050
Masukkan Nama: Hermione
Masukkan Kelas: Gryffindor
Masukkan IPK: 4,0

Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
0. Keluar
Pilih menu: 2
Masukkan NIM: 50403020
Masukkan Nama: Zacky
Masukkan Kelas: TI1E
Masukkan IPK: 3,75
```

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
0. Keluar
Pilih menu: 5
NIM: 20304050, Nama: Hermione, Kelas: Gryffindor, IPK: 4.0
NIM: 50403020, Nama: Zacky, Kelas: TI1E, IPK: 3.75
```

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
0. Keluar
Pilih menu: 7
Masukkan NIM setelah node yang ingin disisipkan: 20304050
Masukkan data mahasiswa yang akan disisipkan:
Masukkan NIM: 12345
Masukkan Nama: Rio
Masukkan Kelas: TI1E
Masukkan IPK: 3,90
Node berhasil disisipkan setelah NIM 20304050

Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
0. Keluar
Pilih menu: 5
NIM: 20304050, Nama: Hermione, Kelas: Gryffindor, IPK: 4.0
NIM: 12345, Nama: Rio, Kelas: TI1E, IPK: 3.9
NIM: 50403020, Nama: Zacky, Kelas: TI1E, IPK: 3.75
```

12.3 Kegiatan Praktikum 2

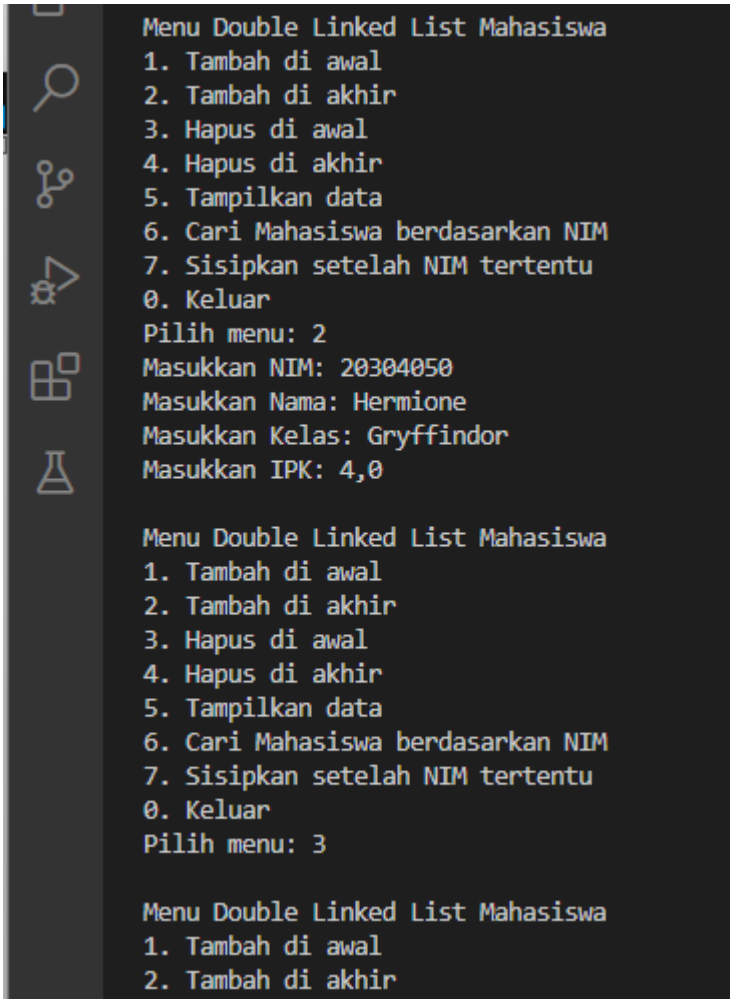
12.3.1 Tahapan Percobaan

Class DoubleLinkedList27 method removeFirst dan removeLast

```
    public void removeFirst() {
        if (isEmpty()) {
            System.out.println("List kosong, tidak bisa dihapus.");
            return;
        }
        if (head == tail) {
            head = tail = null;
        } else {
            head = head.next;
            head.prev = null;
        }
    }


    public void removeLast() {
        if (isEmpty()) {
            System.out.println("List Kosong, tidak bisa dihapus.");
            return;
        }
        if (head == tail) {
            head = tail = null;
        } else {
            tail = tail.prev;
            tail.next = null;
        }
    }
```

Hasil output



12.3.3 Pertanyaan Percobaan

1. Apakah maksud statement berikut pada method **removeFirst()**?
head = head.next;
head.prev = null;

Jawaban :

Untuk memindahkan head ke node yang ada setelahnya, karena node pertama mau dihapus. Setelah itu, kita putuskan hubungan ke node lama dengan membuat prev (penunjuk ke belakang) jadi kosong. Jadi, node lama benar-benar putus dari daftar dan tidak nyambung lagi

2. Modifikasi kode program untuk menampilkan pesan “Data sudah berhasil dihapus. Data yang terhapus adalah … “

Jawaban :

Modifikasi pada class DoubleLinkedList27 pada method removeFirst dan removeLast

```
    public void removeFirst() {
        if (isEmpty()) {
            System.out.println("List kosong, tidak bisa dihapus.");
            return;
        }
        Mahasiswa27 removeData = head.data;
        if (head == tail) {
            head = tail = null;
        } else {
            head = head.next;
            head.prev = null;
        }
        System.out.println("Data sudah berhasil dihapus. Data yang terhapus
adalah:");
        removeData.tampil();
    }


    public void removeLast() {
        if (isEmpty()) {
            System.out.println("List Kosong, tidak bisa dihapus.");
            return;
        }
        Mahasiswa27 removeData = head.data;
        if (head == tail) {
            head = tail = null;
        } else {
            tail = tail.prev;
            tail.next = null;
        }
        System.out.println("Data sudah berhasil dihapus. Data yang terhapus
adalah:");
        removeData.tampil();
    }
```

Hasil Output

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
0. Keluar
Pilih menu: 1
Masukkan NIM: 12345
Masukkan Nama: Zacky
Masukkan Kelas: TI1E
Masukkan IPK: 3,75

Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
0. Keluar
Pilih menu: 2
Masukkan NIM: 54321
Masukkan Nama: Rio
Masukkan Kelas: TI1E
Masukkan IPK: 3,85
```

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
0. Keluar
Pilih menu: 5
NIM: 12345, Nama: Zacky, Kelas: TI1E, IPK: 3.75
NIM: 54321, Nama: Rio, Kelas: TI1E, IPK: 3.85

Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
0. Keluar
Pilih menu: 3
Data sudah berhasil dihapus. Data yang terhapus adalah:
NIM: 12345, Nama: Zacky, Kelas: TI1E, IPK: 3.75
```

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
0. Keluar
Pilih menu: 4
Data sudah berhasil dihapus. Data yang terhapus adalah:
NIM: 54321, Nama: Rio, Kelas: TI1E, IPK: 3.85
```

12.5 Tugas Praktikum

1. Tambahkan fungsi add() pada kelas DoubleLinkedList untuk menambahkan node pada indeks tertentu

Jawaban :

Method add() pada class DoubleLinkedList27

```
    public void add(int index, Mahasiswa27 data) {
    if (index < 0) {
        System.out.println("Indeks tidak boleh negatif.");
        return;
    }
    if (index == 0) {
        addFirst(data);
        return;
    }
    Node27 current = head;
    int count = 0;
    while (current != null && count < index - 1) {
        current = current.next;
        count++;
    }
    if (current == null) {
        System.out.println("Indeks melebihi panjang list. Data akan
ditambahkan di akhir.");
        addLast(data);
        return;
    }
    Node27 newNode = new Node27(data);
    newNode.next = current.next;
    newNode.prev = current;
    if (current.next != null) {
        current.next.prev = newNode;
    } else {
        tail = newNode;
    }
    current.next = newNode;
    System.out.println("Data berhasil ditambahkan pada indeks ke-" +
index);
    }
```

Tambahan pada class DLLMain

```
System.out.println("8. Tambah data pada indeks tertentu");
case 8 -> {
                    System.out.print("Masukkan indeks: ");
                    int index = scan.nextInt();
                    scan.nextLine();
                    Mahasiswa27 mhs = inputMahasiswa(scan);
                    list.add(index, mhs);
                }
```

Hasil Outputnya

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
8. Tambah data pada indeks tertentu
0. Keluar
Pilih menu: 1
Masukkan NIM: 1
Masukkan Nama: Zacky
Masukkan Kelas: TI1E
Masukkan IPK: 3,75

Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
8. Tambah data pada indeks tertentu
0. Keluar
Pilih menu: 2
Masukkan NIM: 2
Masukkan Nama: Rio
Masukkan Kelas: TI1E
Masukkan IPK: 3,9
```

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
8. Tambah data pada indeks tertentu
0. Keluar
Pilih menu: 5
NIM: 1, Nama: Zacky, Kelas: TI1E, IPK: 3.75
NIM: 2, Nama: Rio, Kelas: TI1E, IPK: 3.9

Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
8. Tambah data pada indeks tertentu
0. Keluar
Pilih menu: 8
Masukkan indeks: 1
Masukkan NIM: 3
Masukkan Nama: Orlando
Masukkan Kelas: TI1E
Masukkan IPK: 3,7
Data berhasil ditambahkan pada indeks ke-1
```

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
8. Tambah data pada indeks tertentu
0. Keluar
Pilih menu: 5
NIM: 1, Nama: Zacky, Kelas: TI1E, IPK: 3.75
NIM: 3, Nama: Orlando, Kelas: TI1E, IPK: 3.7
NIM: 2, Nama: Rio, Kelas: TI1E, IPK: 3.9
```

2. Tambahkan removeAfter() pada kelas DoubleLinkedList untuk menghapus node setelah data **key.**

Jawaban :

Method removeAfter() pada class DoubleLinkedList27

```
    public void removeAfter(String keyNim) {
        if (isEmpty()) {
            System.out.println("List kosong, tidak ada yang bisa
dihapus.");
            return;
        }
        Node27 current = head;
        while (current != null && !current.data.nim.equals(keyNim)) {
            current = current.next;
        }
        if (current == null) {
            System.out.println("Node dengan NIM " + keyNim + " tidak
ditemukan.");
            return;
        }
        if (current.next == null) {
            System.out.println("Tidak ada node setelah NIM " + keyNim + "
untuk dihapus.");
            return;
        }
        Node27 nodeToRemove = current.next;
        Mahasiswa27 removeData = nodeToRemove.data;
        if (nodeToRemove == tail) {
            tail = current;
            current.next = null;
        } else {
            current.next = nodeToRemove.next;
            nodeToRemove.next.prev = current;
        }
        System.out.println("Data sudah berhasil dihapus setelah NIM " +
keyNim + ". Data yang terhapus adalah:");
        removeData.tampil();
    }
```

Tambahan pada class DLLMain

```
System.out.println("9. Hapus node setelah NIM tertentu");
case 9 -> {
                    System.out.print("Masukkan NIM setelah node yang akan
dihapus: ");
                    String keyNim = scan.nextLine();
                    list.removeAfter(keyNim);
                }
```

Hasil output

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
8. Tambah data pada indeks tertentu
9. Hapus node setelah NIM tertentu
0. Keluar
Pilih menu: 5
NIM: 1, Nama: Zacky, Kelas: TI1E, IPK: 3.75
NIM: 2, Nama: Rio, Kelas: TI1E, IPK: 3.9

Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
8. Tambah data pada indeks tertentu
9. Hapus node setelah NIM tertentu
0. Keluar
Pilih menu: 9
Masukkan NIM setelah node yang akan dihapus: 1
Data sudah berhasil dihapus setelah NIM 1. Data yang terhapus adala
h:
NIM: 2, Nama: Rio, Kelas: TI1E, IPK: 3.9
```

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
8. Tambah data pada indeks tertentu
9. Hapus node setelah NIM tertentu
0. Keluar
Pilih menu: 1
Masukkan NIM: 1
Masukkan Nama: Zacky
Masukkan Kelas: TI1E
Masukkan IPK: 3,75

Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
8. Tambah data pada indeks tertentu
9. Hapus node setelah NIM tertentu
0. Keluar
Pilih menu: 2
Masukkan NIM: 2
Masukkan Nama: Rio
Masukkan Kelas: TI1E
Masukkan IPK: 3,9
```

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
8. Tambah data pada indeks tertentu
9. Hapus node setelah NIM tertentu
0. Keluar
Pilih menu: 5
NIM: 1, Nama: Zacky, Kelas: TI1E, IPK: 3.75
```

3. Tambahkan fungsi remove() pada kelas DoubleLinkedList untuk menghapus node pada indeks tertentu.

Jawaban:

Method remove() pada class DoubleLinkedList27

```
    public void remove(int index) {
        if (isEmpty()) {
            System.out.println("List kosong, tidak bisa dihapus.");
            return;
        }
        if (index < 0) {
            System.out.println("Indeks tidak boleh negatif.");
            return;
        }
        if (index == 0) {
            removeFirst();
            return;
        }
        Node27 current = head;
        int count = 0;
        while (current != null && count < index) {
            current = current.next;
            count++;
        }
        if (current == null) {
            System.out.println("Indeks melebihi panjang list, tidak ada
data yang dihapus.");
            return;
        }
        if (current == tail) {
            removeLast();
            return;
        }
        Mahasiswa27 removeData = current.data;
        current.prev.next = current.next;
        current.next.prev = current.prev;
        System.out.println("Data sudah berhasil dihapus pada indeks ke-" +
index + ". Data yang terhapus adalah:");
        removeData.tampil();
    }
```

Tambahan pada class DLLMain

```
System.out.println("10. Hapus data pada indeks tertentu");
case 10 -> {
                    System.out.print("Masukkan indeks yang akan dihapus:
");
                    int index = scan.nextInt();
                    scan.nextLine();
                    list.remove(index);
                }
```

Hasil Outputnya

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
8. Tambah data pada indeks tertentu
9. Hapus node setelah NIM tertentu
10. Hapus data pada indeks tertentu
0. Keluar
Pilih menu: 1
Masukkan NIM: 1
Masukkan Nama: Zacky
Masukkan Kelas: TI1E
Masukkan IPK: 3,75
```

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
8. Tambah data pada indeks tertentu
9. Hapus node setelah NIM tertentu
10. Hapus data pada indeks tertentu
0. Keluar
Pilih menu: 2
Masukkan NIM: 2
Masukkan Nama: Rio
Masukkan Kelas: TI1E
Masukkan IPK: 3,5
```

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
8. Tambah data pada indeks tertentu
9. Hapus node setelah NIM tertentu
10. Hapus data pada indeks tertentu
0. Keluar
Pilih menu: 10
Masukkan indeks yang akan dihapus: 1
Data sudah berhasil dihapus. Data yang terhapus adalah:
NIM: 1, Nama: Zacky, Kelas: TI1E, IPK: 3.75
```

4. Tambahkan fungsi getFirst(), getLast() dan getIndex() untuk menampilkan data pada node head, node tail dan node pada indeks tertentu.

Jawaban : Method getFirst(), getLast() dan getIndex() pada class DoubleLinkedList27

```
    public void getFirst() {
    if (isEmpty()) {
        System.out.println("List kosong. Tidak ada data di awal.");
    } else {
        System.out.println("Data pada node pertama:");
        head.data.tampil();
    }
}

    public void getLast() {
        if (isEmpty()) {
            System.out.println("List kosong. Tidak ada data di akhir.");
        } else {
            System.out.println("Data pada node terakhir:");
            tail.data.tampil();
        }
    }

    public void getIndex(int index) {
        if (isEmpty()) {
            System.out.println("List kosong. Tidak ada data.");
            return;
        }
        if (index < 0) {
            System.out.println("Indeks tidak boleh negatif.");
            return;
        }
        Node27 current = head;
        int count = 0;
        while (current != null && count < index) {
            current = current.next;
            count++;
        }
        if (current == null) {
            System.out.println("Indeks melebihi panjang list.");
        } else {
            System.out.println("Data pada indeks ke-" + index + ":");
            current.data.tampil();
        }
    }
```

Tambahan pada class DLLMain

```
System.out.println("11. Tampilkan data pertama");
System.out.println("12. Tampilkan data terakhir");
System.out.println("13. Tampilkan data pada indeks tertentu");
                case 11 -> list.getFirst();
                case 12 -> list.getLast();
                case 13 -> {
                    System.out.print("Masukkan indeks: ");
                    int index = scan.nextInt();
                    scan.nextLine();
                    list.getIndex(index);
                }
```

Hasil Output

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
8. Tambah data pada indeks tertentu
9. Hapus node setelah NIM tertentu
10. Hapus data pada indeks tertentu
11. Tampilkan data pertama
12. Tampilkan data terakhir
13. Tampilkan data pada indeks tertentu
0. Keluar
Pilih menu: 1
Masukkan NIM: 1
Masukkan Nama: Zacky
Masukkan Kelas: TI1E
Masukkan IPK: 3,75
```

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
8. Tambah data pada indeks tertentu
9. Hapus node setelah NIM tertentu
10. Hapus data pada indeks tertentu
11. Tampilkan data pertama
12. Tampilkan data terakhir
13. Tampilkan data pada indeks tertentu
0. Keluar
Pilih menu: 2
Masukkan NIM: 2
Masukkan Nama: Rio
Masukkan Kelas: TI1E
Masukkan IPK: 3,5
```

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
8. Tambah data pada indeks tertentu
9. Hapus node setelah NIM tertentu
10. Hapus data pada indeks tertentu
11. Tampilkan data pertama
12. Tampilkan data terakhir
13. Tampilkan data pada indeks tertentu
0. Keluar
Pilih menu: 11
Data pada node pertama:
NIM: 1, Nama: Zacky, Kelas: TI1E, IPK: 3.75
```

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
8. Tambah data pada indeks tertentu
9. Hapus node setelah NIM tertentu
10. Hapus data pada indeks tertentu
11. Tampilkan data pertama
12. Tampilkan data terakhir
13. Tampilkan data pada indeks tertentu
0. Keluar
Pilih menu: 12
Data pada node terakhir:
NIM: 2, Nama: Rio, Kelas: TI1E, IPK: 3.5
```

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
8. Tambah data pada indeks tertentu
9. Hapus node setelah NIM tertentu
10. Hapus data pada indeks tertentu
11. Tampilkan data pertama
12. Tampilkan data terakhir
13. Tampilkan data pada indeks tertentu
0. Keluar
Pilih menu: 13
Masukkan indeks: 1
Data pada indeks ke-1:
NIM: 2, Nama: Rio, Kelas: TI1E, IPK: 3.5
```

5. tambahkan kode program dan fungsi agar dapat membaca size/ jumlah data pada Double Linked List

Jawaban :

Class DoubleLInkedList memodifikasi beberapa bagian

```
    int size;

    public DoubleLinkedList27() {
        head = null;
        tail = null;
        size = 0;
    }

    public void addFirst(Mahasiswa27 data) {
        Node27 newNode27 = new Node27(data);
        if (isEmpty()) {
            head = tail = newNode27;
        } else {
            newNode27.next = head;
            head.prev = newNode27;
            head = newNode27;
        }
        size++;
    }

    public void addLast(Mahasiswa27 data) {
        Node27 newNode27 = new Node27(data);
        if (isEmpty()) {
            head = tail = newNode27;
        } else {
            tail.next = newNode27;
            newNode27.prev = tail;
            tail = newNode27;
        }
        size++;
    }

    public void inserAfter(String keyNim, Mahasiswa27 data) {
        Node27 current = head;
        while (current != null && !current.data.nim.equals(keyNim)) {
            current = current.next;
        }
        if (current == null) {
            System.out.println("Node dengan NIM " + keyNim + " tidak ditemukan.");
            return;
        }
        Node27 newNode27 = new Node27(data);
        if (current == tail) {
            current.next = newNode27;
            newNode27.prev = current;
            tail = newNode27;
        } else {
            newNode27.next = current.next;
            newNode27.prev = current;
            current.next.prev = newNode27;
            current.next = newNode27;
        }
        size++;
        System.out.println("Node berhasil disisipkan setelah NIM " + keyNim);
    }

    public void removeLast() {
        if (isEmpty()) {
            System.out.println("List Kosong, tidak bisa dihapus.");
            return;
        }
        Mahasiswa27 removeData = head.data;
        if (head == tail) {
            head = tail = null;
        } else {
            tail = tail.prev;
            tail.next = null;
        }
        size--;
        System.out.println("Data sudah berhasil dihapus. Data yang terhapus adalah:");
        removeData.tampil();
    }
```

Tambahan pada class DLLMain

```
System.out.println("14. Lihat jumlah data (size)");
case 14 -> {
                    System.out.println("Jumlah data dalam linked list: " +
list.getSize());
                }
```

Hasil Output

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
8. Tambah data pada indeks tertentu
9. Hapus node setelah NIM tertentu
10. Hapus data pada indeks tertentu
11. Tampilkan data pertama
12. Tampilkan data terakhir
13. Tampilkan data pada indeks tertentu
14. Lihat jumlah data (size)
0. Keluar
Pilih menu: 1
Masukkan NIM: 1
Masukkan Nama: Zacky
Masukkan Kelas: TI1E
Masukkan IPK: 3,75
```

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
8. Tambah data pada indeks tertentu
9. Hapus node setelah NIM tertentu
10. Hapus data pada indeks tertentu
11. Tampilkan data pertama
12. Tampilkan data terakhir
13. Tampilkan data pada indeks tertentu
14. Lihat jumlah data (size)
0. Keluar
Pilih menu: 2
Masukkan NIM: 2
Masukkan Nama: Rio
Masukkan Kelas: TI1E
Masukkan IPK: 3,5
```

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan setelah NIM tertentu
8. Tambah data pada indeks tertentu
9. Hapus node setelah NIM tertentu
10. Hapus data pada indeks tertentu
11. Tampilkan data pertama
12. Tampilkan data terakhir
13. Tampilkan data pada indeks tertentu
14. Lihat jumlah data (size)
0. Keluar
Pilih menu: 14
Jumlah data dalam linked list: 2
```