Nama : Wahyudi Satriawan Hamid

Nim : 244107020137

Verifikasi Hasil Percobaan 1

```
==== MENU DOUBLE LINKED LIST ====
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: 1
NIM   : 20304050
Nama  : Hermione
Prodi : Gryffindor
IPK : 4.0

==== MENU DOUBLE LINKED LIST ====
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: 5
Isi Double Linked List:
NIM   : 20304050
Nama  : Hermione
Prodi : Gryffindor
IPK   : 4.0
----------------------
```

Pertanyaan 1

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

   Jawaban:

   - **Single Linked List**: Setiap node hanya punya satu arah (pointer next) menuju node selanjutnya.
   - **Double Linked List**: Setiap node memiliki dua arah (prev dan next), sehingga traversal bisa dilakukan maju dan mundur.

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

   Jawaban:

   - `next` digunakan untuk menunjuk ke node setelahnya
   - `prev` digunakan untuk menunjuk ke node sebelumnya

     Keduanya digunakan agar traversal bisa dilakukan ke dua arah (bidirectional).

3.
```
public DoubleLinkedList01() {
    head = null;
    tail = null;
}
```
Konstruktor tersebut digunakan untuk menginisialisasi linked list dalam keadaan kosong, yaitu dengan:

- `head = null` → tidak ada node awal
- `tail = null` → tidak ada node akhir

4.
```
if (isEmpty()) {
    head = tail = newNode;
```
Jika linked list kosong, maka node baru akan menjadi satu-satunya node, sehingga head dan tail sama-sama menunjuk ke newNode.

5. arti statement head.prev = newNode?

   Jawaban:

   Statement tersebut digunakan untuk menghubungkan node lama (yang sebelumnya head) ke node baru sebagai node sebelumnya (prev).

   Artinya, newNode jadi node sebelum head, dan head harus tahu itu.

6. Tambahkan pengecekan isEmpty() di awal fungsi:
7. current.next.prev = newNode;

   Baris ini mengatur agar node setelah current menunjuk kembali ke newNode sebagai node sebelumnya.

   Ini menjaga koneksi dua arah (prev dan next) dalam double linked list.

Verifikasi Hasil Percobaan 2

```
==== MENU DOUBLE LINKED LIST ====
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: 2
NIM   : 20304050
Nama  : Hermione
Prodi : Gryffindor
IPK : 4.0

==== MENU DOUBLE LINKED LIST ====
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: 3
```

Pertanyaan 2

1. maksud statement berikut

   ```
   head = head.next;
   ```

   ```
   head.prev = null;
   ```

   - head = head.next;

     Memindahkan pointer head ke node berikutnya, menghapus node pertama secara logika.

   - head.prev = null;

     Setelah head baru ditentukan, kita hapus koneksi ke node sebelumnya (yang sudah terhapus), agar tidak ada referensi kebelakang lagi.

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

```
public void removeFirst() {
    if (!isEmpty()) {
        Mahasiswa25 dataDihapus = head.data;

        if (size == 1) {
            head = tail = null;
        } else {
            head = head.next;
            head.prev = null;
        }
        size--;

        System.out.println("Data sudah berhasil dihapus.");
        System.out.println("Data yang terhapus adalah:");
        dataDihapus.tampil();
    } else {
        System.out.println("Linked List kosong.");
    }
}
```

Tugas

1.
```java
public void add(int index, Mahasiswa25 data) {
    if (index < 0 || index > size) {
        System.out.println("Index di luar jangkauan!");
        return;
    }

    if (index == 0) {
        addFirst(data);
    } else if (index == size) {
        addLast(data);
    } else {
        Node25 current = head;
        for (int i = 0; i < index; i++) {
            current = current.next;
        }
        Node25 newNode = new Node25(current.prev, data, current);
        current.prev.next = newNode;
        current.prev = newNode;
        size++;
    }
}
```

2.
```java
public void removeAfter(String nimKey) {
    Node25 current = head;
    while (current != null && !current.data.nim.equals(nimKey)) {
        current = current.next;
    }

    if (current == null || current.next == null) {
        System.out.println("Tidak ada node setelah NIM tersebut atau NIM tidak ditemukan.");
    } else {
        Node25 nodeToDelete = current.next;
        if (nodeToDelete == tail) {
            removeLast();
        } else {
            current.next = nodeToDelete.next;
            nodeToDelete.next.prev = current;
            size--;
            System.out.println("Data setelah NIM " + nimKey + " berhasil dihapus.");
        }
    }
}
```

3.
```java
public void remove(int index) {
    if (index < 0 || index >= size) {
        System.out.println("Index di luar jangkauan!");
        return;
    }

    if (index == 0) {
        removeFirst();
    } else if (index == size - 1) {
        removeLast();
    } else {
        Node25 current = head;
        for (int i = 0; i < index; i++) {
            current = current.next;
        }
        current.prev.next = current.next;
        current.next.prev = current.prev;
        size--;
        System.out.println("Data pada index " + index + " berhasil dihapus.");
    }
}
```

4.
```java
public void getFirst() {
    if (!isEmpty()) {
        System.out.println("Data pertama:");
        head.data.tampil();
    } else {
        System.out.println("List kosong.");
    }
}

public void getLast() {
    if (!isEmpty()) {
        System.out.println("Data terakhir:");
        tail.data.tampil();
    } else {
        System.out.println("List kosong.");
    }
}

public void getIndex(int index) {
    if (index < 0 || index >= size) {
        System.out.println("Index di luar jangkauan!");
        return;
    }
    Node25 current = head;
    for (int i = 0; i < index; i++) {
        current = current.next;
    }
    System.out.println("Data pada index " + index + ":");
    current.data.tampil();
}
```

5.
```java
public int getSize() {
    return size;
}
```