LAPORAN PRAKTIKUM ALGORITMA DASAR PEMROGRAMAN

JOBSHEET 12

SHAFIQA NABILA MAHARANI KHOIRUNNISA

244107020221

TI – 1B

12.2.1 Percobaan 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
0. Keluar
Pilih menu: 1
Masukkan NIM: 1111
Masukkan Nama: Fika
Masukkan Kelas: 1B
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
0. Keluar
Pilih menu: 5
NIM: Fika, Nama: 1111, Kelas: 1B, IPK: 4.0
```

Pertanyaan :

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

- Single Linked List hanya punya pointer next, jadi hanya bisa ditelusuri maju. Double Linked List punya next dan prev, jadi bisa ditelusuri dua arah (maju dan mundur).

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

- next menunjuk node setelahnya, prev menunjuk node sebelumnya. Digunakan agar data bisa ditelusuri dua arah.

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

- Untuk menginisialisasi head dan tail menjadi null, artinya linked list masih kosong saat pertama dibuat.

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

- Kode ini menghubungkan node baru ke depan dari node yang sudah ada, lalu update head supaya menunjuk ke node baru.

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

- Untuk menghubungkan node yang sebelumnya menjadi node kedua dengan node baru yang sekarang jadi head.

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

- Tambahkan pengecekan di awal method print() seperti:

```
if (isEmpty()) {
    System.out.println("Linked list kosong.");
}
```

7. Pada insertAfter(), apa maksud dari kode berikut ? current.next.prev = newNode;

- Agar node setelah current menghubungkan prev-nya ke newNode, jadi rantai dua arah tetap tersambung.

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

```
case 7: {
System.out.print(s:"Masukkan NIM yang dicari: ");
String keyNim = scan.nextLine();

Mahasiswa20 mhs = inputMahasiswa(scan);
list.insertAfter(keyNim, mhs);
break;
```

12.3 Kegiatan Praktikum 2

```
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. Tambah data setelah NIM tertentu
0. Keluar
Pilih menu: 2
Masukkan NIM: 1111
Masukkan Nama: Fika
Masukkan Kelas: 1B
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. Tambah data setelah NIM tertentu
0. Keluar
Pilih menu: 3

Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
```

12.3.3 Pertanyaan Percobaan

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

- Baris pertama memindahkan head ke node berikutnya (menghapus node paling depan), Baris kedua memutus hubungan node baru dengan node sebelumnya agar tidak terhubung lagi ke node yang dihapus.

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

```
if (!isEmpty()) {
    Mahasiswa01 mhs = head.data;
    head = head.next;
    if (head != null) head.prev = null;
    System.out.println("Data sudah berhasil dihapus. Data yang terhapus adalah:");
    mhs.tampil();
}
```

12.5 Tugas Praktikum

```
public void add(Mahasiswa20 data, int index) {
    if (index < 0 || index > size()) {
        System.out.println(x:"Indeks tidak valid.");    Replace this use of System.out by a logger.
        return;
    } if (index == 0) {    Move this "if" to a new line or add the missing "else". [+1 location]
        addFirst(data);
        return;
    } if (index == size()) {    Move this "if" to a new line or add the missing "else". [+1 location]
    addLast(data);
    return;
    }

    Node20 current = head;
    for (int i = 0; i < index; i++) {
        current = current.next;
    }
```

```java
public void removeAfter(String keyNim) {
    Node20 current = head;
    while (current != null && !current.data.nim.equals(keyNim)) {
        current = current.next;
    }

    if (current != null && current.next != null) {
        Node20 toDelete = current.next;
        current.next = toDelete.next;
    if (toDelete.next != null) {
        toDelete.next.prev = current;
    } else {
        tail = current;
    }
    System.out.println("Data setelah " + keyNim + " berhasil dihapus: ");    Replace this use of System.out by a logger.
    toDelete.data.tampil();
    } else {
        System.out.println("Data setelah " + keyNim + " tidak ditemukan.");    Replace this use of System.out by a logger.
    }
}
```

```java
public void remove(int index) {
    if (index < 0 || index >= size()) {
        System.out.println(x:"Indeks tidak valid.");    Replace this use of System.out by a logger.
        return;
    } if (index == 0) {    Move this "if" to a new line or add the missing "else". [+1 location]
        removeFirst();
        return;
    } if (index == size() - 1) {    Move this "if" to a new line or add the missing "else". [+1 location]
        removeLast();
        return;
    }

    Node20 current = head;
    for (int i = 0; i < index; i++) {
        current = current.next;
    }

    current.prev.next = current.next;
    current.next.prev = current.prev;
    System.out.println(x:"Data berhasil dihapus:");    Replace this use of System.out by a logger.
    current.data.tampil();
}
```

```java
public Mahasiswa20 getFirst() {
    return head != null ? head.data : null;
}

public Mahasiswa20 getLast() {
    return tail != null ? tail.data : null;
}

public Mahasiswa20 get(int index) {
    if (index < 0 || index >= size()) return null;
        Node20 current = head;    This line will not be executed conditionally; only the first line of this 2-line
    for (int i = 0; i < index; i++) {
        current = current.next;
    }
    return current.data;
}

public int size() {
    int count = 0;
    Node20 current = head;
    while (current != null) {
        count++;
        current = current.next;
    }
    return count;
}
```