LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN STRUKTUR DATA
JOBSHEET 12

Disusun Oleh :

Nama : Nawaf Azril Annaufal

Nim : 244107020047

Kelas : TI 1E

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLINEMA
2025

Percobaan 1:

a. Hasil penulisan kode program class Mahasiswa21.java

```
package Praktikum_ASD.Jobsheet12.Percobaan;
public class Mahasiswa21 {

    String nim, nama, kelas;
    double ipk;

    public Mahasiswa21(String nim, String nama, String kelas, double ipk) {
        this.nim = nim;
        this.nama = nama;
        this.kelas = kelas;
        this.ipk = ipk;
    }

      void tampilInformasi() {
        System.out.printf("%-10s %-10s %-5s %.1f\n", nama, nim, kelas, ipk);

    }
}
```

b. Hasil penulisn kode program class Node21.java

```
package Praktikum_ASD.Jobsheet12.Percobaan;
public class Node21 {

    Mahasiswa21 data;
    Node21 prev;
    Node21 next;

    public Node21(Mahasiswa21 data, Node21 prev, Node21 next) {
        this.data = data;
        this.prev = null;
        this.next = null;
    }
}
```

c. Hasil penulisan kode program DoubleLinkedList21.java

```
package Praktikum_ASD.Jobsheet12.Percobaan;

public class DoubleLinkedList21 {

    Node21 head;
    Node21 tail;

    public DoubleLinkedList21() {
        head = null;
        tail = null;
    }

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

    public void addFirst(Mahasiswa21 data) {
        Node21 newNode = new Node21(data, null, null);
        if (isEmpty()) {
            head = tail = newNode;
        } else {
            newNode.next = head;
            head.prev = newNode;
            head = newNode;
        }
    }

    public void addLast(Mahasiswa21 data) {
        Node21 newNode = new Node21(data, null, null);
        if(isEmpty()) {
            head = tail = newNode;
        } else {
            tail.next = newNode;
            newNode.prev = tail;
            tail = newNode;
        }
    }

    public void insertAfter(String keyNim, Mahasiswa21 data) {
        Node21 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;
        }
```

```
  Node21 newNode = new Node21(data, null, null);

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

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

    public void print() {
        Node21 current = head;
        while (current != null) {
            current.data.tampilInformasi();
            current = current.next;
        }
    }

    public Node21 search(String nim) {
        Node21 current = head;
        while(current != null) {
            if (current.data.nim.equals(nim)) {
                return current;
            }

            current = current.next;
        }
        return null;
    }

}
```

d. Hasil penulisan kode program class DDLMain21.java

```
package Praktikum_ASD.Jobsheet12.Percobaan;
import java.util.Scanner;

public class DLLMain21 {
    static Scanner sc = new Scanner(System.in);

    static public Mahasiswa21 inputMahasiswa() {
        System.out.print("Masukkan NIM: ");
        String nim = sc.nextLine();
        System.out.print("Masukkan Nama: ");
        String nama = sc.nextLine();
        System.out.print("Masukkan Kelas: ");
        String kelas = sc.nextLine();
        System.out.print("Masukkan IPK: ");
        double ipk = sc.nextDouble();
        sc.nextLine();
        return new Mahasiswa21(nim, nama, kelas, ipk);
    }

    public static void main(String[] args) {
        DoubleLinkedList21 list = new DoubleLinkedList21();
        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("6. Cari Mahasiswa berdasarkan NIM");
            System.out.println("0. Keluar");
            System.out.print("Pilih Menu: ");
            pilihan = sc.nextInt();
            sc.nextLine();

            switch (pilihan) {
                case 1 -> {
                    Mahasiswa21 mhs = inputMahasiswa();
                    list.addFirst(mhs);
                }

                case 2 -> {
                    Mahasiswa21 mhs = inputMahasiswa();
                    list.addLast(mhs);
                }
```

```
 case 3 -> System.out.println();
                case 4 -> System.out.println();
                case 5 -> list.print();
                case 6 -> {
                    System.out.println("Masukkan NIM yang dicari: ");
                    String nim = sc.nextLine();
                    Node21 found = list.search(nim);
                    if(found != null) {
                        System.out.println("Data Ditemukan");
                        found.data.tampilInformasi();
                    } else {
                        System.out.println("Data tidak ditemukan");
                    }
                }

                default -> System.out.println("Pilihan tidak valid!");
            }
        } while (pilihan != 0);
    }
}
```

e. Hasil run kode program

```
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 nama tertentu
0. Keluar
Pilih Menu: 1
Masukkan NIM: 1234
Masukkan Nama: Adi
Masukkan Kelas: TI1E
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 nama tertentu
0. Keluar
Pilih Menu: 5
Adi         1234        TI1E  4.0
```

Pertanyaan:

1. Jelaskan perbedaan singke linked list dengan double linked list!
2. Perhatikan class Node21, di dalamnya terdapat atribut next dan prev, untuk apakah atribut tersebut?
3. Perhatikan konstraktor pada class DoubleLinkedList, apa kegunaan dari konstruktor tersebut?

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

4. Pada mthod addFirst(), apa makud dari kode berikut?

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

5. Perhatikan pada method addFirst(), Apakah arti statement head.prev = newNode?
6. Modifikasi pada fungsi print() agar dapat menampilkan warning/ pesan bahwa linkedlist masih dalam kondisi kosong
7. Pada insertAfter(), apa maksud dari kode berikut?
   Current.next.prev = newNode;
8. Modifikasi menu pilihan dan switch-case agar fungsi insertAfter() masuk ke dalam menu pilihan dan berjalan dengan baik!

Jawab:

1. Perbedaan paling jelas antara Single dan Double Linked List terdapat pada penggunakan pointernya, dimaan SLL masih menggunakan next saja sebagai penujuk ke data berikutnya, sedangan DLL sudah menggunakan 2 buah pointer yaitu next dan prev yang membuat DLL lebih fleksibel daripada SSL karena dalam mengakses data, DDL dapat dilakukan dengan cara akses dari depan ke belakang atau dari belakang ke depan, sedangkan SSL hanya dari depan ke belakang saja.

2. Next dan Prev dalam DLL disebut sebagai pointer / object references yang menunjuk referensi objek lain / node lain dalam kasus ini next menunjuk node setelah current sedangkan Prev menunjuk node sebelum current.

3. Construktor tersebut digunakan untuk memastikan bahwa linked list pada saat pertama kali dibuat dalam keadaan kosong (head = tail = null) agar linked list dapat digunakan secara aman sejak awal.

4. Jika method isEmpty = true atau kondisi linked list saat ini masih kosong maka head dan tail akan di set ke data yang baru secara langsung.

5. Head.prev = newNode digunakan untuk menghubungkan node lama yang sekarang berada di depan (head) agar tau bahwa ada node baru sebelumnya yaitu newNode

6. Hasil modifikasi agar method print() dapat menampilkan warning saat data pada linked list masih kosong

```
public void print() {
        if(isEmpty()) {
            System.out.println("Data masih kosong");
            return;
        }
        Node21 current = head;
        while (current != null) {
            current.data.tampilInformasi();
            current = current.next;
        }
    }
```

7. Maksud dari kode tersebut adalah node yang tadinya setelah current sekarang harus tau bahwa node sebelum dirinya ada newNode
8. Modifikasi agar method insertAfter(); dapat digunakan

```
case 7 -> {
                System.out.println("Masukkan nama setelah siapa akan disisipkan");
                String key = sc.nextLine();
                list.insertAfter(key, inputMahasiswa());
            }
```

Percobaan 2:

a. Hasil penulisan kode program class DoubleLinkedList.java

```
public void removeFirst() {
        if (isEmpty()) {
            System.out.println("Data masih kosong");
        } else if (head == null) {
            head = tail = null;
        } else {
            head = head.next;
            head.prev = null;
        }
    }

    public void removeLast() {
        if (isEmpty()) {
            System.out.println("Data masih kosong");
        } else if (head == tail) {
            head = tail = null;
        } else {
            tail = tail.prev;
            tail.next = null;
        }
```

b. Hasil run kode program

```
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 nama tertentu
0. Keluar
Pilih Menu: 4

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 nama tertentu
0. Keluar
Pilih Menu: 5
Data masih kosong
```

```
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 nama tertentu
0. Keluar
Pilih Menu: 5
Heru        1234        TI1E  4.0
Lisa        145         SI1A  3.8

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 nama tertentu
0. Keluar
Pilih Menu: 3
```

Pertanyaan:

1. Apakah maksud statement berikut pada method **removeFirst()**?
   head = head.next;
   head.prev = null;
2. Modifikasi kode program untuk menampilkan pesan “Data sudah berhasil dihapus. Data yang terhapus adalah … “

Jawab:

1. Statement head = head.next digunakan untuk melewati head saat ini ke head berikutnya (menunjuk head yang baru) agar head yang lama tidak ditunjuk lagi (terhapus), lalu statemennt head.prev = null digunakan untuk memastikan supaya tidak ada node lain sebelum head yang baru.
2. Hasil modifikasi kode program agar mthod removelast dan removefirst dapat menampilkan data yang akan dihapus.

```
  public void removeFirst() {
        if (isEmpty()) {
            System.out.println("Data masih kosong");
        } else if (head == null) {
            System.out.println("Data sudah berhasil dihapus, data yang
terhapus adalah: "
            + head.data.nim + " " + head.data.nama + " " + head.data.kelas
+ " " + head.data.ipk);
            head = tail = null;
        } else {
            System.out.println("Data sudah berhasil dihapus, data yang
terhapus adalah: "
            + head.data.nim + " " + head.data.nama + " " + head.data.kelas
+ " " + head.data.ipk);
            head = head.next;
            head.prev = null;
        }
    }

    public void removeLast() {
        if (isEmpty()) {
            System.out.println("Data masih kosong");
        } else if (head == tail) {
            System.out.println("Data sudah berhasil dihapus, data yang
terhapus adalah: "
            + tail.data.nim + " " + tail.data.nama + " " + tail.data.kelas
+ " " + tail.data.ipk);
            head = tail = null;
        } else {
            System.out.println("Data sudah berhasil dihapus, data yang
terhapus adalah: "
            + tail.data.nim + " " + tail.data.nama + " " + tail.data.kelas
+ " " + tail.data.ipk);
            tail = tail.prev;
            tail.next = null;
        }
    }
```

Tugas:

1. Menambahkan method add() yang digunakan untuk menambahkan data pada index tertrntu.

```
public void add(int index, Mahasiswa21 input) {
      if (index < 0) {
          System.out.println("Index salah");
      }

      if (index == 0) {
          addFirst(input);
          return;
      }

      Node21 current = head;
      int currentIndex = 0;

      while (current != null && currentIndex < index) {
          current = current.next;
          currentIndex++;
      }

      if(current == null) {
          addLast(input);
      } else {
          Node21 newNode = new Node21(input, current.prev, current);

          if (current.prev != null) {
              current.prev.next = newNode;
          }
          current.prev = newNode;

           if (newNode.prev == null) {
               head = newNode;
           }
      }
  }
```

2. Menambahkan method removeAfter() untuk menghapus data setelah key, disini key menggunakan NIM.

```
  public void removeAfter(String keyNim) {
        Node21 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 + "
yang bisa dihapus");
            return;
        }

        Node21 nodeToRemove = current.next;

        current.next = nodeToRemove.next;

        if (nodeToRemove.next != null) {
            nodeToRemove.prev.next = current;
        } else {
            tail = current;
        }

        System.out.println("Node setelah NIM " + keyNim + " berhasil
dihapus. data yang dihapus: "
        + nodeToRemove.data.nim + " - " + nodeToRemove.data.nama);
    }
```

3. Menambahkan method remove() untuk menghapus data pada index tertentu.

```
    public void remove(int index) {
        if(isEmpty()) {
            System.out.println("Data masih kosong");
        }

        if (index < 0) {
            System.out.println("index yang dimasukkan salah");
            return;
        }

        if (index == 0) {
            removeFirst();
            return;
        }

        Node21 current = head;
        int currentIndex = 0;

        while (current != null && currentIndex < index) {
            current = current.next;
            currentIndex++;
        }

        if(current == null) {
            System.out.println("Index melebihi jumlah node");
            return;
        }

        if (current.next != null) {
            current.next.prev = current.prev;
        } else {
            tail = current.prev;
        }

        if(current.prev != null) {
            current.prev.next = current.next;
        }

        System.out.println("Node pada index " + index + " berhasil
dihapus. Data: "
        + current.data.nim + " - " + current.data.nama);



    }
```

4. Menambahkan method getFirst(), getLast(), dan getIndex() yang digunakan untuk menampilkan data pertama, terakhir dan data dari index tertentu.

```
 public void getFirst() {
        if (isEmpty()) {
            System.out.println("Data masih kosong");
        } else {
            System.out.println("Data Pertama: " + head.data.nim + " - " +
head.data.nama);
        }
    }

    public void getLast() {
        if (isEmpty()) {
            System.out.println("Data masih kodong");
        } else {
            System.out.println("Data Terakhir: " + tail.data.nim + " - " +
tail.data.nama);
        }
    }

    public void getIndex(int index) {
        if (isEmpty()) {
            System.out.println("Data masih kosong");
            return;
        }

        if (index < 0) {
            System.out.println("Index tidak valid");
            return;
        }

        Node21 current = head;
        int currentIndex = 0;

        while (current != null && currentIndex < index) {
            current = current.next;
            currentIndex++;
        }
        if(current == null) {
            System.out.println("Index melebihi panjang list");
        } else {
            System.out.println("Data pada index " + index + ": " +
            current.data.nim + " - " + current.data.nama);
        }

    }
```

5. Yang pertama dilakukan adalah menambahkan size pada constructor DoubleLinkedList agar dapat menyimpan nilai size berupa integer.

```
public DoubleLinkedList21() {
        head = null;
        tail = null;
        size = 0;
    }
```

Lalu menambahkan size++ pada setiap method yang menambahkan data dan menambahkan size-- pada setiap method yang menghapus data, setelah itu membuat method getSize() untuk menampilkan size saat ini.

```
public void getSize() {
       if (size == 0) {
           System.out.println("Data masih kosong");
       } else {
           System.out.println("Jumlah data sekarang adalah: " + size);
       }

   }
```