PRAKTIKUM ALGORITMA DAN STRUKTUR DATA

JOBSHEET 4

NAMA : DIMAS ADI BAYU SAMUDRA

KELAS : 1A

NO. ABSEN : 08

NIM : 2341720169

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2024

Kode Node

```
public class Node {
    int data;
    Node next;

    Node (int nilai, Node berikutnya){
        data = nilai;
        next = berikutnya;
    }
}
```

Kode SingleLinkedList

```
public class SingleLinkedList {
    Node head, tail;

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

    void print() {
        if (!isEmpty()) {
            Node tmp = head;
            System.out.print("Isi Linked List ");
            while (tmp != null) {
                System.out.print(tmp.data + "\t");
                tmp = tmp.next;
            }
            System.out.println();
        } else {
            System.out.println("Linked List kosong");
        }
    }

    void addFirst(int input) {
        Node ndInput = new Node(input, null);
        if (isEmpty()) {
            head = ndInput;
            tail = ndInput;
        } else {
            ndInput.next = head;
            head = ndInput;
        }
    }

    void addLast(int input) {
        Node ndInput = new Node(input, null);
        if (isEmpty()) {
            head = ndInput;
            tail = ndInput;
```

```
        } else {
            tail.next = ndInput;
            tail = ndInput;
        }
    }

    void inserAfter(int key, int input) {
        Node ndInput = new Node(input, null);
        Node temp = head;
        while (temp != null) {
            if (temp.data == key) {
                ndInput.next = temp.next;
                temp.next = ndInput;
                if (ndInput.next == null) {
                    tail = ndInput;
                }
                break;
            }
            temp = temp.next;
        }
    }

    void inserAt(int index, int input) {
        Node ndInput = new Node(input, null);
        if (index < 0) {
            System.out.println("Index cannot be negative.");
            return;
        }
        if (index == 0) {
            addFirst(input);
        } else {
            Node temp = head;
            for (int i = 0; temp != null && i < index - 1; i++) {
                temp = temp.next;
            }
            if (temp == null) {
                System.out.println("Index out of bounds.");
            } else {
                ndInput.next = temp.next;
                temp.next = ndInput;
                if (ndInput.next == null) {
                    tail = ndInput;
                }
            }
        }
    }

}
```

Kode main

```
public class SLLMain {
    public static void main(String[] args) {
        SingleLinkedList singLL = new SingleLinkedList();
```

```
        singLL.print();
        singLL.addFirst(890);
        singLL.print();
        singLL.addLast(760);
        singLL.print();
        singLL.addFirst(700);
        singLL.print();
        singLL.inserAfter(700, 999);
        singLL.print();
        singLL.inserAt(3, 833);
        singLL.print();
    }
}
```

Hasil run

```
Linked List kosong
Isi Linked List 890
Isi Linked List 890     760
Isi Linked List 700     890     760
Isi Linked List 700     999     890     760
Isi Linked List 700     999     890     833     760
```

Pertanyaan

1. Mengapa hasil compile kode program di baris pertama menghasilkan “Linked List Kosong”?

2. Jelaskan kegunaan variable temp secara umum pada setiap method!

3. Perhatikan class SingleLinkedList, pada method insertAt Jelaskan kegunaan kode berikut

```
if(temp.next.next==null) tail=temp.next;
```

Jawaban

1. Di karenakan ada beberapa metod yang isinya harus di ubah , seperti pada metod addFisrt(); menghapus tail = ndInput; dalam else, Memindahkan pernyataan tail = ndInput; ke dalam blok else, Menambahkan else setelah kondisi if (index == 0).

2. print(): Digunakan untuk mengunjungi setiap node dalam linked list dan mencetak nilainya. addFirst(int input): Tidak menggunakan temp. addLast(int input): Tidak menggunakan temp. insertAfter(int key, int input): Digunakan untuk mencari node dengan nilai key agar dapat memasukkan node baru setelahnya. insertAt(int index, int input): Digunakan untuk mencapai posisi di mana elemen baru akan dimasukkan.

3. Jika if terpenuhi maka nilai tail di ubah menjadi nilai temp.next .

Percobaan 2

Kode tambahan SingleLingkedList

```
public class SingleLinkedList {

    Node head, tail;

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

    void print() {
        if (isEmpty()) {
            Node tmp = head;
            System.out.print("Isi Linked List:\t");
            while (tmp != null) {
                System.out.print(tmp.data + "\t");
                tmp = tmp.next;
            }
            System.out.println("");
        } else {
            System.out.println("Linked List kosong");
        }
    }

    void addFirst(int input) {
        Node ndInput = new Node(input, null);
        if (isEmpty()) {
            ndInput.next = head;
            head = ndInput;
        } else {
            head = ndInput;
            tail = ndInput;
        }
    }

    void addLast(int input) {
        Node ndInput = new Node(input, null);
        if (isEmpty()) {
            tail.next = ndInput;
            tail = ndInput;
        } else {
            head = ndInput;
            tail = ndInput;
        }
    }

    void insertAfter(int key, int input) {
        Node ndInput = new Node(input, null);
        Node temp = head;
        while (temp != null) {
            if (temp.data == key) {
```

```
                ndInput.next = temp.next;
                temp.next = ndInput;
                if (ndInput.next != null) {
                    tail = ndInput;
                }
                break;
            }
            temp = temp.next;
        }
    }

    void insertAt(int index, int input) {
        if (index < 0) {
            System.out.println("Perbaiki logikanya!" + "Kalau indeksnya -1
bagaimana???");
        } else if (index == 0) {
            addFirst(input);
        } else {
            Node temp = head;
            for (int i = 0; i < index - 1; i++) {
                temp = temp.next;
            }
            temp.next = new Node(input, temp.next);
            if (temp.next.next == null) {
                tail = temp.next;
            }
        }
    }

    int getData(int index) {
        Node tmp = head;
        for (int i = 0; i < index - 1; i++) {
            tmp = tmp.next;
        }
        return tmp.next.data;
    }

    int indexOf(int key) {
        Node tmp = head;
        int index = 0;
        while (tmp != null && tmp.data != key) {
            tmp = tmp.next;
            index++;
        }
        if (tmp == null) {
            return 1;
        } else {
            return index;
        }
    }

    void removeFirst() {
        if (!isEmpty()) {
```

```
            System.out.println("Linked List Masih Kosong," + "Tidak Dapat
Dihapus");
        } else if (head == tail) {
            head = tail = null;
        } else {
            head = head.next;
        }
    }

    void removeLast() {
        if (!isEmpty()) {
            System.out.println("Linked List Masih Kosong." + "Tidak Dapat
Dihapus");
        } else if (head == tail) {
            head = tail = null;
        } else {
            Node temp = head;
            while (temp.next == null) {
                temp = temp.next;
            }
            temp.next = null;
            tail = temp.next;
        }
    }

    void remove(int key) {
        if (!isEmpty()) {
            System.out.println("Linked List Masih Kosong." + "Tidak Dapat
Dihapus");
        } else {
            Node temp = head;
            while (temp != null) {
                if (temp.data == key && temp == head) {
                    removeFirst();
                    break;
                } else if (temp.next.data == key) {
                    temp.next = temp.next.next;
                    if (temp.next == null) {
                        tail = temp;
                    }
                    break;
                }
                temp = temp.next;
            }
        }
    }

    void removeAt(int index){
        if (index == 0) {
            removeFirst();
        } else {
            Node temp = head ;
            for (int i = 0; i < index; i++) {
```

```
                temp = temp.next;
            }
            temp.next = temp.next.next;
            if (temp.next == null) {
                tail = temp;
            }
        }
    }

}
```

Kode tambahan main

```
public class SLLMain {
    public static void main(String[] args) {
        SingleLinkedList singLL = new SingleLinkedList();
        singLL.print();
        singLL.addFirst(890);
        singLL.print();
        singLL.addLast(760);
        singLL.print();
        singLL.addFirst(700);
        singLL.print();
        singLL.insertAfter(700, 999);
        singLL.print();
        singLL.insertAt(3, 833);
        singLL.print();

        System.out.println("Data pada indeks ke-1 " + singLL.getData(1));
        System.out.println("Data 3 berada pada indeks ke- " +
singLL.indexOf(760));

        singLL.remove(999);
        singLL.print();
        singLL.removeAt(0);
        singLL.print();
        singLL.removeFirst();
        singLL.print();
        singLL.removeLast();
        singLL.print();
    }
}
```

Pertanyaan

1. Mengapa digunakan keyword break pada fungsi remove? Jelaskan!

2. Jelaskan kegunaan kode dibawah pada method remove

```
else if (temp.next.data == key) {
  temp.next = temp.next.next;
```

Jawaban

1. Keyword break digunakan dalam metode remove(int key) untuk menghentikan loop saat suatu kondisi terpenuhi. Ini dilakukan untuk menghindari pengecekan berlebihan dan mempercepat eksekusi kode.

2. Jika kondisi else if terpenuhi, artinya elemen yang akan dihapus ditemukan di tengah daftar. Oleh karena itu, referensi temp.next diperbarui untuk mengarah ke elemen berikutnya setelah elemen yang akan dihapus.

Tugas 1

Kode Node

```
package Tugas1;
public class NodeTugas108 {
    String nama;
    int nim;
    NodeTugas108 next;

    public NodeTugas108(int nim, String nama, NodeTugas108 berikutnya) {
        this.nim = nim;
        this.nama = nama;
        next = berikutnya;
    }

    public NodeTugas108(int nim, String nama) {
        this.nim = nim;
        this.nama = nama;
    }
}
```

Kode Linked List

```
package Tugas1;

public class LinkedList08 {
    NodeTugas108 head, tail;

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

    void print() {
        if (!isEmpty()) {
            NodeTugas108 tmp = head;
            System.out.println("Isi Linked List:");
            int counter = 1;
            while (tmp != null) {
                System.out.println("Mhs" + counter + "\tNIM: " + tmp.nim +
"\tNama: " + tmp.nama);
                tmp = tmp.next;
```

```
                counter++;
            }
        } else {
            System.out.println("Linked List kosong");
        }
    }

    void addFirst(int nim, String nama) {
        NodeTugas108 ndInput = new NodeTugas108(nim, nama, null);
        if (isEmpty()) {
            head = ndInput;
            tail = ndInput;
        } else {
            ndInput.next = head;
            head = ndInput;
        }
    }

    void addLast(int nim, String nama) {
        NodeTugas108 ndInput = new NodeTugas108(nim, nama, null);
        if (isEmpty()) {
            head = ndInput;
            tail = ndInput;
        } else {
            tail.next = ndInput;
            tail = ndInput;
        }
    }

    void insertAfter(int key, int nim, String nama) {
        NodeTugas108 ndInput = new NodeTugas108(nim, nama, null);
        NodeTugas108 temp = head;
        while (temp != null) {
            if (temp.nim == key) {
                ndInput.next = temp.next;
                temp.next = ndInput;
                if (ndInput.next == null) {
                    tail = ndInput;
                }
                break;
            }
            temp = temp.next;
        }
    }

    void insertAt(int index, int nim, String nama) {
        if (index < 0) {
            System.out.println("Perbaiki logikanya! Kalau indeksnya -1
bagaimana???");
        } else if (index == 0) {
            addFirst(nim, nama);
        } else {
            NodeTugas108 temp = head;
```

```
            for (int i = 0; i < index - 1; i++) {
                temp = temp.next;
            }
            temp.next = new NodeTugas108(nim, nama, temp.next);
            if (temp.next.next == null) {
                tail = temp.next;
            }
        }
    }
}
```

Kode main

```
package Tugas1;

import java.util.Scanner;

public class SLLTugas108 {
        public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        LinkedList08 singLL = new LinkedList08();

        singLL.print();
        singLL.addFirst(112, "prita");
        singLL.print();
        singLL.addLast(113, "yusuf");
        singLL.print();
        singLL.addFirst(111, "anton");
        singLL.print();
        singLL.insertAfter(113, 114, "sari");
        singLL.print();
        singLL.insertAt(3, 115, "doni");
        singLL.print();
        }
}
```

Hasil Run

```
Linked List kosong
Isi Linked List:
Mhs1    NIM: 112        Nama: prita
Isi Linked List:
Mhs1    NIM: 112        Nama: prita
Mhs2    NIM: 113        Nama: yusuf
Isi Linked List:
Mhs1    NIM: 111        Nama: anton
Mhs2    NIM: 112        Nama: prita
Mhs3    NIM: 113        Nama: yusuf
Isi Linked List:
Mhs1    NIM: 111        Nama: anton
Mhs2    NIM: 112        Nama: prita
Mhs3    NIM: 113        Nama: yusuf
Mhs4    NIM: 114        Nama: sari
Isi Linked List:
Mhs1    NIM: 111        Nama: anton
Mhs2    NIM: 112        Nama: prita
Mhs3    NIM: 113        Nama: yusuf
Mhs4    NIM: 115        Nama: doni
Mhs5    NIM: 114        Nama: sari
```

Tugas 2

Kode Node

```
package Tugas2;
public class NodeTugas208 {
    String nama;
    int nim;
    NodeTugas208 next;

    public NodeTugas208(int nim, String nama, NodeTugas208 berikutnya) {
        this.nim = nim;
        this.nama = nama;
        next = berikutnya;
    }

    public NodeTugas208(int nim, String nama) {
        this.nim = nim;
        this.nama = nama;
    }
}
```

Kode AntrianMahasiswa

```
package Tugas2;


public class AntrianMahasiswa08 {
    NodeTugas208 head, tail;

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

    void print() {
        if (!isEmpty()) {
            NodeTugas208 tmp = head;
            System.out.println("Isi Linked List:");
            int counter = 1;
            while (tmp != null) {
                System.out.println("Mhs" + counter + "\tNIM: " + tmp.nim +
"\tNama: " + tmp.nama);
                tmp = tmp.next;
                counter++;
            }
        } else {
            System.out.println("Linked List kosong");
        }
    }

    void addFirst(int nim, String nama) {
        NodeTugas208 ndInput = new NodeTugas208(nim, nama, null);
        if (isEmpty()) {
            head = ndInput;
```

```
            tail = ndInput;
        } else {
            ndInput.next = head;
            head = ndInput;
        }
    }

    void addLast(int nim, String nama) {
        NodeTugas208 ndInput = new NodeTugas208(nim, nama, null);
        if (isEmpty()) {
            head = ndInput;
            tail = ndInput;
        } else {
            tail.next = ndInput;
            tail = ndInput;
        }
    }

    void insertAfter(int key, int nim, String nama) {
        NodeTugas208 ndInput = new NodeTugas208(nim, nama, null);
        NodeTugas208 temp = head;
        while (temp != null) {
            if (temp.nim == key) {
                ndInput.next = temp.next;
                temp.next = ndInput;
                if (ndInput.next == null) {
                    tail = ndInput;
                }
                break;
            }
            temp = temp.next;
        }
    }

    void insertAt(int index, int nim, String nama) {
        if (index < 0) {
            System.out.println("Perbaiki logikanya! Kalau indeksnya -1
bagaimana???");
        } else if (index == 0) {
            addFirst(nim, nama);
        } else {
            NodeTugas208 temp = head;
            for (int i = 0; i < index - 1; i++) {
                temp = temp.next;
            }
            temp.next = new NodeTugas208(nim, nama, temp.next);
            if (temp.next.next == null) {
                tail = temp.next;
            }
        }
    }

    public void enqueue(int nim, String nama) {
```

```
    NodeTugas208 ndInput = new NodeTugas208(nim, nama, null);
    if (isEmpty()) {
        head = ndInput;
        tail = ndInput;
        } else {
        tail.next = ndInput;
        tail = ndInput;
        }
    }

    public NodeTugas208 dequeue() {
        if (isEmpty()) {
            System.out.println("Antrian kosong.");
            return null;
        }
        NodeTugas208 temp = head;
        head = head.next;
        if (head == null) {
            tail = null;
        }
        return temp;
    }
}
```

Kode Main

```
package Tugas2;

import java.util.Scanner;

public class SLLTugas208 {
        public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        LinkedList08 singLL = new LinkedList08();

        singLL.print();
        singLL.addFirst(112, "prita");
        singLL.print();
        singLL.addLast(113, "yusuf");
        singLL.print();
        singLL.addFirst(111, "anton");
        singLL.print();
        singLL.insertAfter(113, 114, "sari");
        singLL.print();
        singLL.insertAt(3, 115, "doni");
        singLL.print();
        }
}
```

Hasil Run

```
Linked List kosong
Isi Linked List:
Mhs1    NIM: 112        Nama: prita
Isi Linked List:
Mhs1    NIM: 112        Nama: prita
Mhs2    NIM: 113        Nama: yusuf
Isi Linked List:
Mhs1    NIM: 111        Nama: anton
Mhs2    NIM: 112        Nama: prita
Mhs3    NIM: 113        Nama: yusuf
Isi Linked List:
Mhs1    NIM: 111        Nama: anton
Mhs2    NIM: 112        Nama: prita
Mhs3    NIM: 113        Nama: yusuf
Mhs4    NIM: 114        Nama: sari
Isi Linked List:
Mhs1    NIM: 111        Nama: anton
Mhs2    NIM: 112        Nama: prita
Mhs3    NIM: 113        Nama: yusuf
Mhs4    NIM: 115        Nama: doni
Mhs5    NIM: 114        Nama: sari
```