PRAKTIKUM ALGORITMA DAN STRUKTUR DATA

JOBSHEET PERTEMUAN KE-9

NAMA : ALVINO VALERIAN D.R

KELAS : 1A

NO. ABSEN : 05

NIM : 2341720027

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2024

2.1.1 Verifikasi Hasil Percobaan

```
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
```

2.1.2 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.Hasil compile program menampilkan "Linked List Kosong" pada baris pertama karena pada saat dijalankan, kondisi isEmpty() mengembalikan nilai true, yang berarti head adalah null, menandakan bahwa linked list tidak memiliki elemen, atau dalam kata lain, kosong.

2.Variabel temp digunakan sebagai pointer atau penunjuk sementara dalam perulangan atau iterasi melalui linked list. Dalam konteks program ini, temp digunakan untuk mengakses dan memanipulasi node-node dalam linked list. Pada setiap iterasi, temp akan menunjuk ke node saat ini, dan kemudian bergeser ke node berikutnya jika perlu.

3.Kode if (temp.next.next == null) dalam metode InsertAt digunakan untuk memeriksa apakah node yang ditambahkan akan menjadi node terakhir dalam linked list setelah penyisipan dilakukan. Jika ya, maka tail perlu diperbarui untuk menunjuk ke node terakhir yang baru ditambahkan. Jadi, ketika kondisi tersebut terpenuhi, tail diperbarui untuk menunjuk ke node yang baru ditambahkan, yaitu temp.next.

2.2.2 Verifikasi Hasil Percobaan

```
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
Data Pada indeks ke-1=999
Data 3 berada pada indeks ke-4
Linked List masih kosong, Tidak dapat dihapus
Isi Linked List 700     999     890     833     760
Isi Linked List 999     890     833     760
Isi Linked List 890     833     760
Isi Linked List 890     833

D:\alvino\Semester 2\Prak algorita & struktur data\josbheet9>
```

2.2.3 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 fungsi remove untuk menghentikan iterasi melalui linked list setelah operasi penghapusan dilakukan. Jika tidak ada break, iterasi akan terus berlanjut bahkan setelah operasi penghapusan dilakukan, yang dapat menyebabkan kesalahan atau perubahan yang tidak diinginkan pada linked list.

2. Kode else if (temp.next.data == key) dalam metode remove bertanggung jawab untuk menemukan node yang berisi nilai yang sama dengan key yang diberikan. Ketika node tersebut ditemukan, maka node tersebut dihapus dari linked list dengan mengatur next dari node sebelumnya untuk melewati node yang akan dihapus, yaitu temp.next, dan mengarahkan ke node setelahnya, yaitu temp.next.next. Dengan demikian, node yang sesuai dengan kriteria dihapus dari linked list.

JAWABAN

1. Keyword break digunakan dalam fungsi remove untuk menghentikan iterasi melalui linked list setelah operasi penghapusan dilakukan. Jika tidak ada break, iterasi akan terus berlanjut bahkan setelah operasi penghapusan dilakukan, yang dapat menyebabkan kesalahan atau perubahan yang tidak diinginkan pada linked list.

2. Kode else if (temp.next.data == key) dalam metode remove bertanggung jawab untuk menemukan node yang berisi nilai yang sama dengan key yang diberikan. Ketika node tersebut ditemukan, maka node tersebut dihapus dari linked list dengan mengatur next dari node sebelumnya untuk melewati node yang akan dihapus, yaitu temp.next, dan mengarahkan ke node setelahnya, yaitu temp.next.next. Dengan demikian, node yang sesuai dengan kriteria dihapus dari linked list.

TUGAS1

MAIN

```
package tugas1;

/**

 * SLLMain

 */

public class SLLMain {



    public static void main(String[] args) {
```

```
        SingleLinkedList singLL=new SingleLinkedList ();
        singLL.addFirst("paul", 2345003);
        singLL.print();
        singLL.addLast("Tio",2345004 );
        singLL.print();
        singLL.addFirst("Vidi", 2345001);
        singLL.print();
        singLL.addLast("Nabeel", 2345002);
        singLL.print();
        singLL.InsertAfter("Vino", 2345002, 2345005);
        singLL.print();



    }

}
```

```
package tugas1;

public class SingleLinkedList {
    Node head,tail;

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

    void print(){
        if (!isEmpty()) {
            Node tmp = head;
            System.out.println("Isi Linked List: ");
            int i=1;
            while (tmp != null) {
```

```
                System.out.println("Mahasiswa ke-"+i);
                System.out.println("Nama:"+tmp.nama +"-" +
"Nim:"+tmp.nim+"\t");
                // System.out.print(tmp.nim + "\t");
                tmp = tmp.next;
                i++;
            }
            System.out.println("");
        } else {
            System.out.println("Linked List Kosong");
        }
    }


    void addFirst(String nama,int input){
        Node ndInput = new Node(nama, input, head);

        if (isEmpty()) {
            head = ndInput;
            tail = ndInput;
        }else{
            ndInput.next = head;
            head = ndInput;
        }
    }

    void addLast(String nama,int input){
        Node ndInput = new Node(nama,input, null);

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

```
        }else{
            tail.next = ndInput;
            tail = ndInput;
        }
    }
    void InsertAfter(String nama, int key, int input) {
        Node ndInput = new Node(nama, input, null);
        Node temp = head;

        while (temp != null) {
            if (temp.nim == key) { // Memeriksa apakah NIM node saat
ini sama dengan kunci
                ndInput.next = temp.next;
                temp.next = ndInput;
                if (ndInput.next == null) {
                    tail = ndInput;
                }
                break;
            }
            temp = temp.next;
        }
    }

    void InsertAt(String nama,int index, int input){
        Node ndInput = new Node(nama,input, null);
        if (index > 0) {
            Node temp = head;
            for (int i = 0; i < index-1; i++) {
                temp = temp.next;
            }
            ndInput.next = temp.next;
            temp.next = ndInput;
            if (temp.next.next == null) {
```

```
                tail = temp.next;
            }
        } else if (index == 0) {
            addFirst(nama,input);
        }else if (index == -1) {
            System.out.println("Linked List masih kosong");
        }
    }
    int getdata(int index){
        Node tmp = head;
        for (int i = 0; i < index-1; i++) {
            tmp = tmp.next;
        }
        return tmp.next.nim;
    }
    int indexOf(int key){
        Node tmp = head;
        int index = 0;
        while (tmp != null && tmp.nim != 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.next == null) {
            head = tail = null;
        }else{
            Node temp = head;
            while (temp.next != tail) {
                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.nim != key && temp == head) {
                    removeFirst();
                    break;
                }else if (temp.next.nim == key) {
                    temp.next = temp.next.next;
                    if (temp.next == null) {
```

```
                        tail = temp;
                    }
                    break;
                }
                temp = temp.next;
            }
        }
    }
    public void removeAt(int index) {
        if (index == 0) {
            removeFirst();
        }else{
            Node temp = head;
            for (int i = 0; i < index -1; i++) {
                temp = temp.next;
            }
            temp.next = temp.next.next;
            if (temp.next == null) {
                tail = temp;
            }
        }
    }
    }
```

```
package tugas1;


public class Node {
    String nama;
    int nim;
    Node next;


    Node(String nama,int nilai, Node berikutnya){
```

```
        this.nama=nama;
        nim = nilai;
        next = berikutnya;
    }

}
```

```
Isi Linked List:
Mahasiswa ke-1
Nama:Vidi-Nim:2345001
Mahasiswa ke-2
Nama:paul-Nim:2345003
Mahasiswa ke-3
Nama:Tio-Nim:2345004
Mahasiswa ke-4
Nama:Nabeel-Nim:2345002
Mahasiswa ke-5
Nama:Vino-Nim:2345005


D:\alvino\Semester 2\Prak algorita & struktur data\josbheet9>
```

TUGAS2

```
package tugas2;


public class Node {
    String nama;
    int nim;
    Node next;


    Node(String nama,int nilai, Node berikutnya){
        this.nama=nama;
        nim = nilai;
        next = berikutnya;
    }


}
```

```
package tugas2;
/**
 * SLLMain
 */
public class SLLMain {



    public static void main(String[] args) {
        SingleLinkedList singll = new  SingleLinkedList();
        System.out.println("---------Antrian Mahasiswa---------");
       singll.add("Pajri", 1234501);
       singll.add("Vino", 1234502);
       singll.add("Nabeel", 1234503);
       singll.add("Gwido", 1234504);
       singll.add("Tio", 1234505);

       singll.print();
       singll.remove();
       singll.print();
    }

}
```

```
package tugas2;


public class SingleLinkedList {
    Node head,tail;


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

```
    void print(){
        if (!isEmpty()) {
            Node tmp = head;
            System.out.println("Isi Linked List: ");
            int i=1;
            while (tmp != null) {
                System.out.println("Mahasiswa ke-"+i);
                System.out.println("Nama:"+tmp.nama +"-" +
"Nim:"+tmp.nim+"\t");
                // System.out.print(tmp.nim + "\t");
                tmp = tmp.next;
                i++;
            }
            System.out.println("");
        } else {
            System.out.println("Linked List Kosong");
        }
    }

    void add(String nama,int input){
        Node ndInput = new Node(nama,input, null);

        if (isEmpty()) {
            head = ndInput;
            tail  = ndInput;

        }else{
            tail.next = ndInput;
            tail = ndInput;
        }
    }
    void remove(){
        if (isEmpty()) {
```

```
            System.out.println("Linked List masih kosong, Tidak
dapat dihapus");
        } else if (head == tail) {
            head = tail = null;
        } else {
            head = head.next;
        }
    }


    }
```

```
---------Antrian Mahasiswa---------
Isi Linked List:
Mahasiswa ke-1
Nama:Pajri-Nim:1234501
Mahasiswa ke-2
Nama:Vino-Nim:1234502
Mahasiswa ke-3
Nama:Nabeel-Nim:1234503
Mahasiswa ke-4
Nama:Gwido-Nim:1234504
Mahasiswa ke-5
Nama:Tio-Nim:1234505
```

```
Isi Linked List:
Mahasiswa ke-1
Nama:Vino-Nim:1234502
Mahasiswa ke-2
Nama:Nabeel-Nim:1234503
Mahasiswa ke-3
Nama:Gwido-Nim:1234504
Mahasiswa ke-4
Nama:Tio-Nim:1234505
```