NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Linked List (Jobsheet 9)

PERCOBAAN 1

```java
package P11.StrukturData;

public class Node {
    int data;
    Node next;

    public Node(int nilai, Node berikutnya) {
        this.data = nilai;
        this.next = berikutnya;
    }

}

```

```java
package P11.StrukturData;

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();
    }
}

```

NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Linked List (Jobsheet 9)

```java
package P11.StrukturData;

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, head);

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

    void insertAt(int index, int input) {
        if (index < 0) {
            System.out.println("Indeks salah");
        } else if (index == 0) {
            addFirst(input);
        } else {
            Node temp = head;
            for (int i = 0; i < index - 1; i++) {
                if (temp.next == null)
                    break;
                temp = temp.next;
            }
            temp.next = new Node(input, temp.next);
            if (temp.next.next == null) {
                tail = temp.next;
            }
        }

    }
}
```

```
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
```

NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Linked List (Jobsheet 9)

Pertanyaan

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

   Jawab:

```
singLL.print();
singLL.addFirst(input:890);
singLL.print();
singLL.addlast(input:760);
singLL.print();
singLL.addFirst(input:700);
singLL.print();
singLL.insertAfter(key:700, input:999);
singLL.print();
singLL.insertAt(index:3, input:833);
singLL.print();
```

   Baris pertama menghasilkan “Linked List Kosong” karena pada linked list tersebut belum diisi sebuah data dan pemanggilan print dilakukan sebelum pengisian data sehingga output program baris pertama adalah “Linked List Kosong”.

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

   Jawab:

   Variabel temp digunakan untuk menampung node sementara.

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

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

   Jawab:

   Code tersebut adalah pengecekan kondisi yang digunakan ketika kondisi di dalam if terpenuhi maka nilai tail akan diperbarui menjadi temp.next

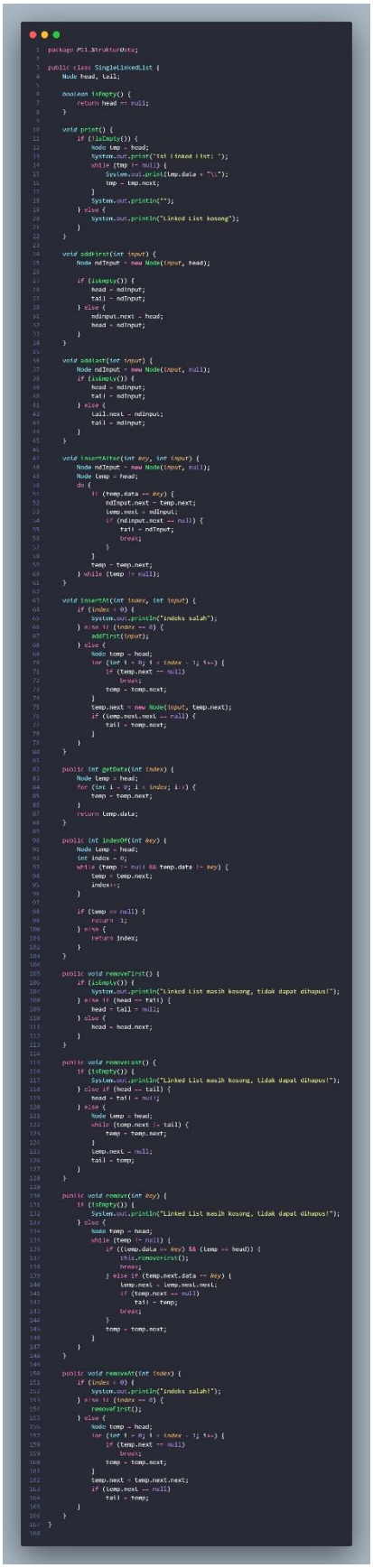
NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Linked List (Jobsheet 9)

PERCOBAAN 2

```java
package P11.StrukturData;

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();
    }
}
```

```
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
Isi Linked List: 700    890     833     760
Isi Linked List: 890    833     760
Isi Linked List: 833    760
Isi Linked List: 833
```

NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Linked List (Jobsheet 9)



```java
package Mi1.StrukturDuts;

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, head);

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

    void insertAt(int index, int input) {
        if (index < 0) {
            System.out.println("Indeks salah");
        } else if (index == 0) {
            addFirst(input);
        } else {
            Node temp = head;
            for (int i = 0; i < index - 1; i++) {
                if (temp.next == null)
                    break;
                temp = temp.next;
            }
            temp.next = new Node(input, temp.next);
            if (temp.next.next == null) {
                tail = temp.next;
            }
        }
    }

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

    public int indexOf(int key) {
        Node temp = head;
        int index = 0;
        while (temp != null && temp.data != key) {
            temp = temp.next;
            index++;
        }

        if (temp == null) {
            return -1;
        } else {
            return index;
        }
    }

    public 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;
        }
    }

    public 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 != tail) {
                temp = temp.next;
            }
            temp.next = null;
            tail = temp;
        }
    }

    public 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)) {
                    this.removeFirst();
                    break;
                } else if (temp.next.data == key) {
                    temp.next = temp.next.next;
                    if (temp.next == null)
                        tail = temp;
                    break;
                }
                temp = temp.next;
            }
        }
    }

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

NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Linked List (Jobsheet 9)

PERTNYAAN

1. Mengapa digunakan keyword break pada fungsi remove? Jelaskan!
   Jawba:
   Keyword break pada fungsi remove digunakan untuk menghentikan iterasi,karena jika dihapus, iterasi akan terus berjalan walaupun node yang dicari sudah ditemukan
2. Jelaskan kegunaan kode dibawah pada method remove

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

   Jawab:
   Kode tersebut digunakan untuk pengecekan kondisi Ketika node yang dicari sama dengan node saat ini (temp.next). Kemudian kode tersebut melakukan penghapusan node dengan cara mengubah pointer dari node temp langsung menuju ke node yang berada seteah node yang akan dihapus.

LINK GITHUB:
https://github.com/RoyW12/AlgoritmaStrukturData_1G_28/tree/main/src/P11/StrukturData