NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Double Linked List (Jobsheet 10)

Percobaan 1

```
Linked Lists Kosong
Size : 0
================================
7       3       4
berhasil diisi
Size : 3
================================
7       40      3       4
berhasil diisi
Size : 4
================================
Linked Lists Kosong
Size : 0
```

```java
package P12;

public class DoubleLinkedListMain {
    public static void main(String[] args) throws Exception {
        DoubleLinkedList dll = new DoubleLinkedList();
        dll.print();
        System.out.println("Size : " + dll.size());
        System.out.println("================================");
        dll.addFirst(3);
        dll.addLast(4);
        dll.addFirst(7);
        dll.print();

        System.out.println("Size : " + dll.size());
        System.out.println("================================");
        dll.add(40, 1);
        dll.print();
        System.out.println("Size : " + dll.size());
        System.out.println("================================");
        dll.clear();
        dll.print();
        System.out.println("Size : " + dll.size());

    }
}
```

NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Double Linked List (Jobsheet 10)

```java
package P12;

public class DoubleLinkedList {
    Node head;
    int size;

    DoubleLinkedList(Node head, int size) {
        head = null;
        size = 0;
    }

    DoubleLinkedList() {

    }

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

    void addFirst(int item) {
        if (isEmpty()) {
            head = new Node(null, item, null);
        } else {
            Node newNode = new Node(null, item, head);
            head.prev = newNode;
            head = newNode;
        }
        size++;
    }

    void addLast(int item) {
        if (isEmpty()) {
            addFirst(item);
        } else {
            Node current = head;
            while (current.next != null) {
                current = current.next;
            }
            Node newNode = new Node(current, item, null);
            current.next = newNode;
            size++;
        }
    }

    void add(int item, int index) throws Exception {
        if (isEmpty()) {
            addFirst(item);
        } else if (index < 0 || index > size) {
            throw new Exception("Nilai indeks di luar batas");
        } else {
            Node current = head;
            int i = 0;
            while (i < index) {
                current = current.next;
                i++;
            }
            if (current.prev == null) {
                Node newNode = new Node(null, item, current);
                current.prev = newNode;
                head = newNode;
            } else {
                Node newNode = new Node(current.prev, item, current);
                current.prev.next = newNode;
                current.prev = newNode;
            }
        }
        size++;
    }

    public int size() {
        return size;
    }

    public void clear() {
        head = null;
        size = 0;
    }

    public void print() {
        if (!isEmpty()) {
            Node tmp = head;
            while (tmp != null) {
                System.out.print(tmp.data + " \t");
                tmp = tmp.next;
            }
            System.out.println("\nberhasil diisi");
        } else {
            System.out.println("Linked Lists Kosong");
        }
    }
}
```

```java
package P12;

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

    Node(Node prev, int data, Node next) {
        this.prev = prev;
        this.data = data;
        this.next = next;

    }
}
```

NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Double Linked List (Jobsheet 10)

PERTANYAAN

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

   Jawab:

   Double linked list mempunyai 2 pointer yaitu next dan prev sedangkan linked list hanya mempunyai 1 pointer yaitu next.

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

   Jawab:

   Atribut next digunakan untuk menunjuk node selanjutnya sedangkan atribut prev digunakan untuk menunjuk node sebelumnya.

3. Perhatikan konstruktor pada class DoubleLinkedLists. Apa kegunaan inisialisasi atribut head dan size seperti pada gambar berikut ini?

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

   Jawab:

   Untuk mendeklarasaikan linked list diawal bahwa nilai head = null, size = 0;

4. Pada method addFirst(), kenapa dalam pembuatan object dari konstruktor class Node prev dianggap sama dengan null? Node newNode = new Node(null, item, head);

   Jawab:

   Ketika pembuatan node baru dengan menggunakan method addFirst, node baru tersebut akan menggantikan head, karena secara konsep nilai prev dari head adalah null.

NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Double Linked List (Jobsheet 10)

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

   Jawab:

   Arti statement tersebut adalah nilai prev dari head mengarah ke newNode atau node baru tersebut. Dan node baru tersebut akan menjadi head.

6. Perhatikan isi method addLast(), apa arti dari pembuatan object Node dengan mengisikan parameter prev dengan current, dan next dengan null? Node newNode = new Node(current, item, null);

   Jawab:

```
while (current.next != null) {
    current = current.next;
}
```

   Kode diatas adalah melakukan traverse, sehingga didapatkan nilai current adalah node terakhir. Lalu nilai next diisi dengan null karena new node tersebut akan menjadi tail, dimana konsep tail adalah nilai nextnya adalah null.

   Nilai prev diisi dengan current dan next diisi dengan null, karena agar new node tersebut mengarah ke nilai node sebelumnya dan mengarah ke null karena new node tersebut menjadi tail

NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Double Linked List (Jobsheet 10)

7. Pada method add(), terdapat potongan kode program sebagai berikut:

```
while (i < index) {
    current = current.next;
    i++;
}
if (current.prev == null) {
    Node newNode = new Node(null, item, current);
    current.prev = newNode;
    head = newNode;
} else {
    Node newNode = new Node(current.prev, item, current);
    newNode.prev = current.prev;
    newNode.next = current;
    current.prev.next = newNode;
    current.prev = newNode;
}
```

jelaskan maksud dari bagian yang ditandai dengan kotak kuning

jawab:

pada perulangan while dilakukan traverse untuk menentukan nilai current. Kemudian dilakukan pengecekan kondisi apakah current.prev == null. Pengecekan kondisi tersebut akan terpenuhi ketika nilai current adalah head karena nilai prev dari head adalah null. Lalu dilakukan pembuatan newNode dengan nilai prev adalah null,next adalah current. Pada baris ketiga yaitu current.prev = newNode agar nilai current atau head mengarah ke node yang baru. Kemudian pada baris terakhir terjadi inisialisasi head = newNode, agar node baru tersebut menjadi head.

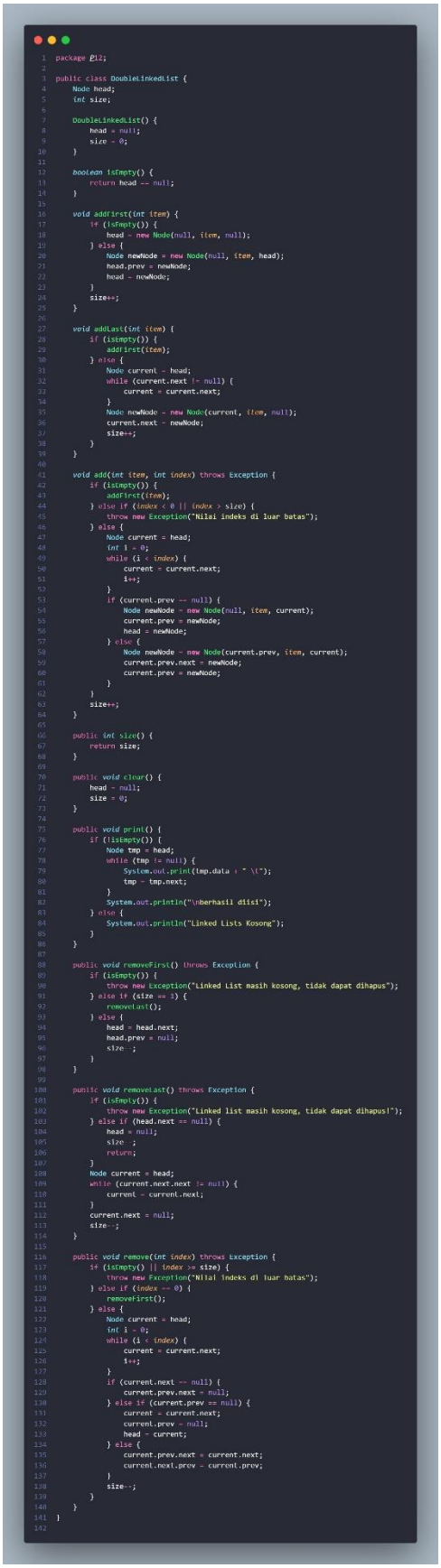
NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Double Linked List (Jobsheet 10)

```
Linked Lists Kosong
Size : 0
================================
7       3       4
berhasil diisi
Size : 3
================================
7       40      3       4
berhasil diisi
Size : 4
================================
Linked Lists Kosong
Size : 0
50      40      10      20
berhasil diisi
Size : 4
================================
40      10      20
berhasil diisi
Size : 3
================================
40      10
berhasil diisi
Size : 2
================================
40
berhasil diisi
Size : 1
```

```java
package P12;

public class DoubleLinkedListMain {
    public static void main(String[] args) throws Exception {
        DoubleLinkedList dll = new DoubleLinkedList();
        dll.print();
        System.out.println("Size : " + dll.size());
        System.out.println("================================");
        dll.addFirst(3);
        dll.addLast(4);
        dll.addFirst(7);
        dll.print();

        System.out.println("Size : " + dll.size());
        System.out.println("================================");
        dll.add(40, 1);
        dll.print();
        System.out.println("Size : " + dll.size());
        System.out.println("================================");
        dll.clear();
        dll.print();
        System.out.println("Size : " + dll.size());

        dll.addLast(50);
        dll.addLast(40);
        dll.addLast(10);
        dll.addLast(20);
        dll.print();
        System.out.println("Size : " + dll.size());
        System.out.println("==================================");
        dll.removeFirst();
        dll.print();
        System.out.println("Size : " + dll.size());
        System.out.println("==================================");
        dll.removeLast();
        dll.print();
        System.out.println("Size : " + dll.size());
        System.out.println("==================================");
        dll.remove(1);
        dll.print();
        System.out.println("Size : " + dll.size());

    }
}
```

```java
package P12;

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

    Node(Node prev, int data, Node next) {
        this.prev = prev;
        this.data = data;
        this.next = next;

    }
}
```

NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Double Linked List (Jobsheet 10)



```java
package P12;

public class DoubleLinkedList {
    Node head;
    int size;

    DoubleLinkedList() {
        head = null;
        size = 0;
    }

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

    void addFirst(int item) {
        if (isEmpty()) {
            head = new Node(null, item, null);
        } else {
            Node newNode = new Node(null, item, head);
            head.prev = newNode;
            head = newNode;
        }
        size++;
    }

    void addLast(int item) {
        if (isEmpty()) {
            addFirst(item);
        } else {
            Node current = head;
            while (current.next != null) {
                current = current.next;
            }
            Node newNode = new Node(current, item, null);
            current.next = newNode;
            size++;
        }
    }

    void add(int item, int index) throws Exception {
        if (isEmpty()) {
            addFirst(item);
        } else if (index < 0 || index > size) {
            throw new Exception("Nilai indeks di luar batas");
        } else {
            Node current = head;
            int i = 0;
            while (i < index) {
                current = current.next;
                i++;
            }
            if (current.prev == null) {
                Node newNode = new Node(null, item, current);
                current.prev = newNode;
                head = newNode;
            } else {
                Node newNode = new Node(current.prev, item, current);
                current.prev.next = newNode;
                current.prev = newNode;
            }
        }
        size++;
    }

    public int size() {
        return size;
    }

    public void clear() {
        head = null;
        size = 0;
    }

    public void print() {
        if (!isEmpty()) {
            Node tmp = head;
            while (tmp != null) {
                System.out.print(tmp.data + " \t");
                tmp = tmp.next;
            }
            System.out.println("\nberhasil diisi");
        } else {
            System.out.println("Linked Lists Kosong");
        }
    }

    public void removeFirst() throws Exception {
        if (isEmpty()) {
            throw new Exception("Linked List masih kosong, tidak dapat dihapus");
        } else if (size == 1) {
            removeLast();
        } else {
            head = head.next;
            head.prev = null;
            size--;
        }
    }

    public void removeLast() throws Exception {
        if (isEmpty()) {
            throw new Exception("Linked list masih kosong, tidak dapat dihapus!");
        } else if (head.next == null) {
            head = null;
            size--;
            return;
        }
        Node current = head;
        while (current.next.next != null) {
            current = current.next;
        }
        current.next = null;
        size--;
    }

    public void remove(int index) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception("Nilai indeks di luar batas");
        } else if (index == 0) {
            removeFirst();
        } else {
            Node current = head;
            int i = 0;
            while (i < index) {
                current = current.next;
                i++;
            }
            if (current.next == null) {
                current.prev.next = null;
            } else if (current.prev == null) {
                current = current.next;
                current.prev = null;
                head = current;
            } else {
                current.prev.next = current.next;
                current.next.prev = current.prev;
            }
            size--;
        }
    }
}
```

NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Double Linked List (Jobsheet 10)

PERTANYAAN

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

   Jawab:

   Head = head.next => pointer head akan diatur ke node berikutnya

   Head.prev = null => pointer prev dari node baru head diatur menjadi null dan menjadi node pertama dalam list dan tidak memiliki node sebelumnya.

2. Bagaimana cara mendeteksi posisi data ada pada bagian akhir pada method removeLast()?

   Jawab:

   Untuk mendeteksi posisi data pada bagian akhir adalah dengan iterasi(traverse).

```
while (current.next.next != null) {
    current = current.next;
}
```

3. Jelaskan alasan potongan kode program di bawah ini tidak cocok untuk perintah remove!

```
Node tmp = head.next;

head.next=tmp.next;
tmp.next.prev=head;
```

   Jawab:

   Jika node yang dihapus adalah node terakhir akan menyebabkan nullpointerexception

NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Double Linked List (Jobsheet 10)

4. Jelaskan fungsi kode program berikut ini pada fungsi remove!

```
current.prev.next = current.next;
current.next.prev = current.prev;
```

Jawab:

Menghubungkan node sebelum dan sesudah node yang akan dihapus dalam double linked list, sehingga daftar tetap terhubung dengan benar setelah node di tengah-tengah dihapus

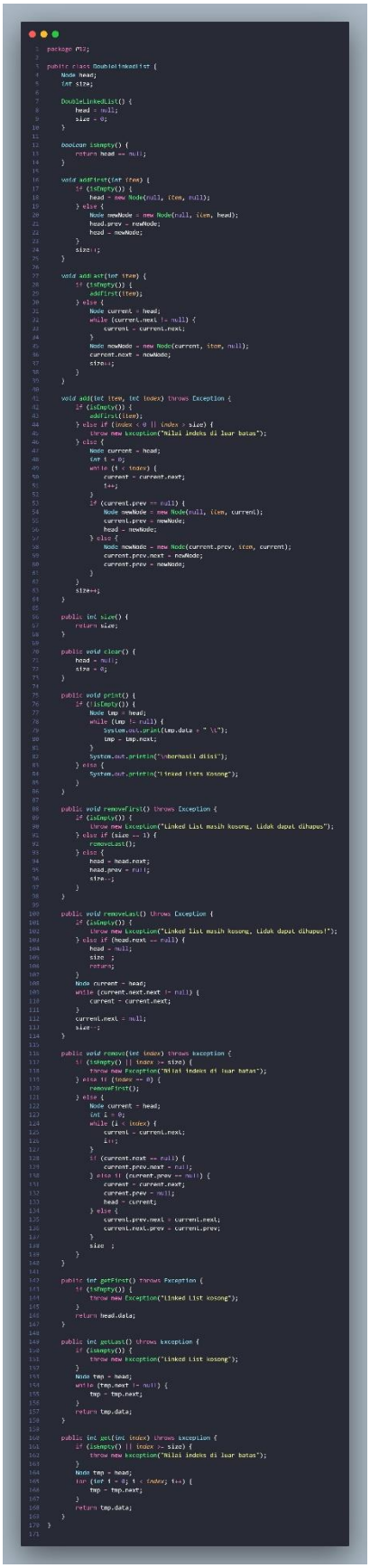
NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Double Linked List (Jobsheet 10)

```
Linked Lists Kosong
Size 0
==============================
7       3       4
berhasil diisi
Size: 3
==============================
7       40      3       4
berhasil diisi
Size: 4
==============================
Data awal pada Linked Lists adalah: 7
Data akhir pada Linked Lists adalah: 4
Data indeks ke-1 pada Linked Lists adalah: 40
```

```java
package P12;

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

    Node(Node prev, int data, Node next) {
        this.prev = prev;
        this.data = data;
        this.next = next;

    }
}

```

```java
package P12;

public class DoubleLinkedListMain {
    public static void main(String[] args) throws Exception {
        DoubleLinkedList dll = new DoubleLinkedList();

        dll.print();
        System.out.println("Size " + dll.size());
        System.out.println("==============================");
        dll.addFirst(3);
        dll.addLast(4);
        dll.addFirst(7);
        dll.print();
        System.out.println("Size: " + dll.size());
        System.out.println("==============================");
        dll.add(40, 1);
        dll.print();
        System.out.println("Size: " + dll.size());
        System.out.println("==============================");
        System.out.println("Data awal pada Linked Lists adalah: " + dll.getFirst());
        System.out.println("Data akhir pada Linked Lists adalah: " + dll.getLast());
        System.out.println("Data indeks ke-1 pada Linked Lists adalah: " + dll.get(1));

    }
}

```

NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Double Linked List (Jobsheet 10)



```
package M12;

public class DoubleLinkedList {
    Node head;
    int size;

    DoubleLinkedList() {
        head = null;
        size = 0;
    }

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

    void addFirst(int item) {
        if (isEmpty()) {
            head = new Node(null, item, null);
        } else {
            Node newNode = new Node(null, item, head);
            head.prev = newNode;
            head = newNode;
        }
        size++;
    }

    void addLast(int item) {
        if (isEmpty()) {
            addFirst(item);
        } else {
            Node current = head;
            while (current.next != null) {
                current = current.next;
            }
            Node newNode = new Node(current, item, null);
            current.next = newNode;
            size++;
        }
    }

    void add(int item, int index) throws Exception {
        if (isEmpty()) {
            addFirst(item);
        } else if (index < 0 || index > size) {
            throw new Exception("Nilai indeks di luar batas");
        } else {
            Node current = head;
            int i = 0;
            while (i < index) {
                current = current.next;
                i++;
            }
            if (current.prev == null) {
                Node newNode = new Node(null, item, current);
                current.prev = newNode;
                head = newNode;
            } else {
                Node newNode = new Node(current.prev, item, current);
                current.prev.next = newNode;
                current.prev = newNode;
            }
        }
        size++;
    }

    public int size() {
        return size;
    }

    public void clear() {
        head = null;
        size = 0;
    }

    public void print() {
        if (!isEmpty()) {
            Node tmp = head;
            while (tmp != null) {
                System.out.print(tmp.data + " \t");
                tmp = tmp.next;
            }
            System.out.println("\nBerhasil diisi");
        } else {
            System.out.println("Linked itsts Kosong");
        }
    }

    public void removeFirst() throws Exception {
        if (isEmpty()) {
            throw new Exception("Linked List masih kosong, tidak dapat dihapus");
        } else if (size == 1) {
            removeLast();
        } else {
            head = head.next;
            head.prev = null;
            size--;
        }
    }

    public void removeLast() throws Exception {
        if (isEmpty()) {
            throw new Exception("Linked list masih kosong, tidak dapat dihapus!");
        } else if (head.next == null) {
            head = null;
            size = 0;
            return;
        }
        Node current = head;
        while (current.next.next != null) {
            current = current.next;
        }
        current.next = null;
        size--;
    }

    public void remove(int index) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception("Nilai indeks di luar batas");
        } else if (index == 0) {
            removeFirst();
        } else {
            Node current = head;
            int i = 0;
            while (i < index) {
                current = current.next;
                i++;
            }
            if (current.next == null) {
                current.prev.next = null;
            } else if (current.prev == null) {
                current = current.next;
                current.prev = null;
                head = current;
            } else {
                current.prev.next = current.next;
                current.next.prev = current.prev;
            }
            size = 0;
        }
    }

    public int getFirst() throws Exception {
        if (isEmpty()) {
            throw new Exception("Linked List kosong");
        }
        return head.data;
    }

    public int getLast() throws Exception {
        if (isEmpty()) {
            throw new Exception("Linked List kosong");
        }
        Node tmp = head;
        while (tmp.next != null) {
            tmp = tmp.next;
        }
        return tmp.data;
    }

    public int get(int index) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception("Nilai indeks di luar batas");
        }
        Node tmp = head;
        for (int i = 0; i < index; i++) {
            tmp = tmp.next;
        }
        return tmp.data;
    }
}
```

NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Double Linked List (Jobsheet 10)

PERTANYAAN

1. Jelaskan method size() pada class DoubleLinkedLists!

   Jawab:

   Untuk mendapatkan nilai size atau method size tersebut mengembalikan nilai size.

2. Jelaskan cara mengatur indeks pada double linked lists supaya dapat dimulai dari indeks ke1!

   Jawab:

   Untuk mengatur indeks pada double linked list (DLL) agar dimulai dari indeks ke-1, perlu dilakukan penyesuaian pada setiap metode yang mengacu pada indeks, terutama metode add(int item, int index), remove(int index), dan get(int index). Saat ini, indeks pada DLL Anda dimulai dari 0, jadi perlu menambahkan atau mengurangi 1 pada parameter indeks saat digunakan.

3. Jelaskan perbedaan karakteristik fungsi Add pada Double Linked Lists dan Single Linked Lists!

   Jawab:

   DLL: Memungkinkan traversal dua arah, sehingga lebih fleksibel untuk operasi yang membutuhkan akses ke elemen sebelumnya.

   SLL: Hanya memungkinkan traversal satu arah, sehingga beberapa operasi mungkin memerlukan traversal tambahan dari awal untuk mengakses elemen sebelumnya.

NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Double Linked List (Jobsheet 10)

4. Jelaskan perbedaan logika dari kedua kode program di bawah ini!

```
public boolean isEmpty(){
    if(size ==0){
        return true;
    } else{
        return false;
    }
}
```

(a)

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

(b)

Jawab**:**

kode a memeriksa menggunakan size apakah nilai size 0 jika benar mengembalikan nilai true jika salah maka sebaliknya yaitu false dengan size adalah ukuran linked list itu sendiri. Kode b memeriksa apakah head bernilai null jika benar fungsi tersebut mengembalikan nilai true dan linked list masih kosong.

LINK GITHUB:
https://github.com/RoyW12/AlgoritmaStrukturData_1G_28