

## JOBSHEET 11

Nama : Farid Aziz Wicaksono

Kelas : TI/1C

Absen : 14

### A. PRAKTIKUM

#### 1. Praktikum1

No	Node.java
1	package minggu11;
2	
3	public class Node {
4	int data;
5	Node prev, next;
6	
7	public Node(Node prev, int data, Node next){
8	this.prev = prev;
9	this.data = data;
10	this.next = next;
11	}
12	}

No	Double Linked List
1	package minggu11;
2	
3	public class doubleLinkedList {
4	Node head;
5	int size;
6	
7	doubleLinkedList() {
8	head = null;
9	size = 0;
10	}
11	

12	boolean isEmpty() {
13	return head == null;
14	}
15	
16	void addFirst(int item) {
17	if (isEmpty()) {
18	head = new Node(null, item, null);
19	} else {
20	Node newNode = new Node(null, item, head);
21	head.prev = newNode;
22	head = newNode;
23	}
24	size++;
25	}
26	
27	void addLast(int item) {
28	if (isEmpty()) {
29	addFirst(item);
30	} else {
31	Node current = head;
32	while (current.next != null) {
33	current = current.next;
34	}
35	Node newNode = new Node(current, item, null);
36	current.next = newNode;
37	size++;
38	}
39	}
40	
41	void add(int item, int index) throws Exception {
42	if (isEmpty()) {

43	addFirst(item);
44	}
45	if (index < 0    index > size) {
46	throw new Exception("Nilai index di luar batas");
47	} else {
48	Node current = head;
49	int i = 0;
50	while (i < index) {
51	current = current.next;
52	}
53	if (current.prev == null) {
54	Node newNode = new Node(null, item, current);
56	current.prev = newNode;
57	head = newNode;
58	} else {
59	Node newNode = new Node(current.prev, item, current);
60	newNode.prev = current.prev;
61	newNode.next = current;
62	current.prev.next = newNode;
63	current.prev = newNode;
64	}
65	}
66	size++;
67	}
68	
68	int size() {
69	return size;
70	}
71	
72	void clear() {
73	head = null;

74	size = 0;
75	}
76	
77	void print() {
78	if (!isEmpty()) {
79	Node tmp = head;
80	while (tmp != null) {
81	System.out.println(tmp.data + "\t");
82	tmp = tmp.next;
83	}
84	System.out.println("berrhasil diisi");
85	} else {
86	System.out.println("Linked list kosong");
87	}
88	}
89	}

No	DoubleLinkedListMain
1	package minggu11;
2	
3	public class doubleLinkedListMain {
4	public static void main(String[] args) {
5	doubleLinkedList dll = new doubleLinkedList();
6	try {
7	dll.print();
8	System.out.println("Size : " + dll.size());
9	
10	System.out.println("=====");
11	dll.addFirst(3);
12	dll.addLast(4);
13	dll.addFirst(7);

14	dll.print();
15	System.out.println("Size : " + dll.size());
16	
17	System.out.println("=====");
18	dll.add(40, 1);
19	dll.print();
20	System.out.println("Size : " + dll.size());
21	
22	System.out.println("=====");
23	dll.clear();
24	dll.print();
25	System.out.println("Size : " + dll.size());
26	
27	System.out.println("=====");
28	} catch (Exception e) {
29	System.out.println(e.getMessage());
30	}
31	}
32	}

Output :

```
run:
PRAKTIKUM 1
Linked list kosong
Size : 0
=====
7
3
4
Berhasil di isi
Size : 3
=====
7
40
3
4
Berhasil di isi
Size : 4
=====
Linked list kosong
Size : 0
```

## 2. Praktikum2

No	DoubleLinkedList
1	void removeFirst() throws Exception {
2	if (isEmpty()) {
3	throw new Exception("Linked List masih Kosong, tidak dapat dihapus!");
4	} else if (size == 1) {
5	removeLast();
6	} else {
7	head = head.next;
8	head.prev = null;
9	size--;
10	}
11	}
12	
13	void removeLast() throws Exception {
14	
15	if (isEmpty()) {
16	throw new Exception("Linked List masih KOsong, tidak dapat dihapus!");
17	}
18	if (head.next == null) {
19	head = null;
20	size--;
21	return;
22	}
23	Node current = head;
24	while (current.next.next != null) {
25	current = current.next;
26	}
27	current.next = null;
28	size--;
29	}

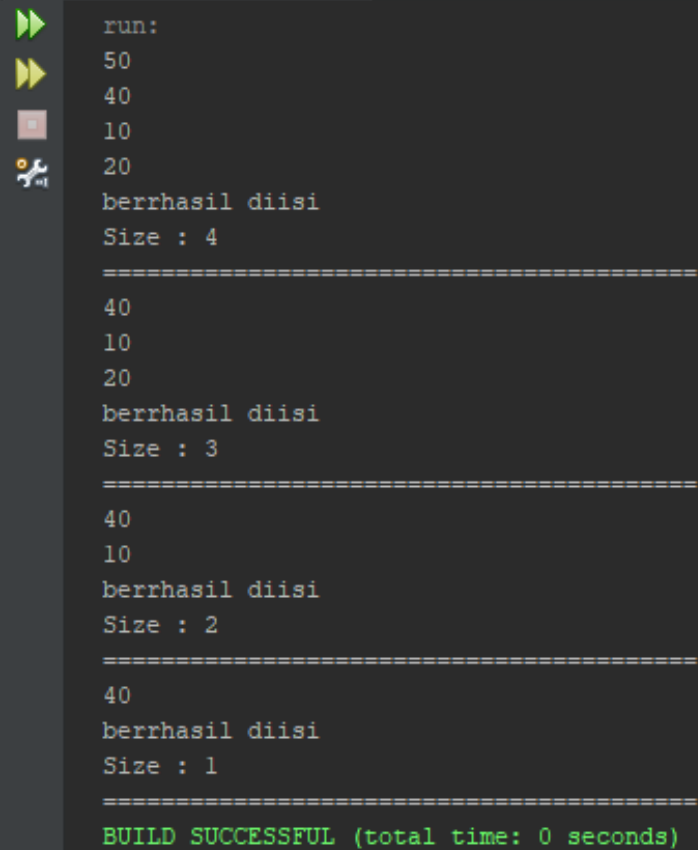
30	
31	void remove(int index) throws Exception {
32	if (isEmpty()    index >= size) {
33	throw new Exception("Nilai indeks di luar batas");
34	}
35	if (index == 0) {
36	removeFirst();
37	} else {
38	Node current = head;
39	int i = 0;
40	while (i < index) {
41	current = current.next;
42	i++;
43	}
44	if (current.next == null) {
45	current.prev.next = null;
46	} else if (current.prev == null) {
47	current = current.next;
48	current.prev = null;
49	head = current;
50	} else {
51	current.prev.next = current.next;
52	current.next.prev = current.prev;
53	}
54	size--;
55	}
56	
57	}

No	DoubleLinkedListMain.java
1	package minggu11;
2	
3	public class doubleLinkedListMain {
4	public static void main(String[] args) {
5	doubleLinkedList dll = new doubleLinkedList();
6	try {
7	dll.addLast(50);
8	dll.addLast(40);
9	dll.addLast(10);
10	dll.addLast(20);
11	dll.print();
12	System.out.println("Size : " + dll.size());
13	
14	System.out.println("=====");
15	dll.removeFirst();
16	dll.print();
17	System.out.println("Size : " + dll.size());
18	
19	System.out.println("=====");
20	dll.removeLast();
21	
22	dll.print();
23	System.out.println("Size : " + dll.size());
24	
25	System.out.println("=====");
26	dll.remove(1);
27	dll.print();
28	System.out.println("Size : " + dll.size());
29	
30	System.out.println("=====");



31	} catch (Exception e) {
32	System.out.println(e.getMessage());
33	}
34	}
35	}

Output :



```
run:
50
40
10
20
berrhasil diisi
Size : 4
=====
40
10
20
berrhasil diisi
Size : 3
=====
40
10
berrhasil diisi
Size : 2
=====
40
berrhasil diisi
Size : 1
=====
BUILD SUCCESSFUL (total time: 0 seconds)
```

### 3. Praktikum3

No	DoubleLinkedList.java
1	public int getFirst() throws Exception {
2	if (isEmpty()) {
3	throw new Exception("Linked List Kosong");
4	}
5	return head.data;
6	}
7	
8	public int getLast() throws Exception {
9	if (isEmpty()) {
10	throw new Exception("Linked List kosong");
11	}
12	Node tmp = head;
13	while (tmp.next != null) {
14	tmp = tmp.next;
15	}
16	return tmp.data;
17	}
18	
19	public int get(int index) throws Exception {
20	if (isEmpty()    index >= size) {
21	throw new Exception("Nilai Index di luar batas");
22	}
23	Node tmp = head;
24	for (int i = 0; i < index; i++) {
25	tmp = tmp.next;
26	}
27	return tmp.data;
28	}

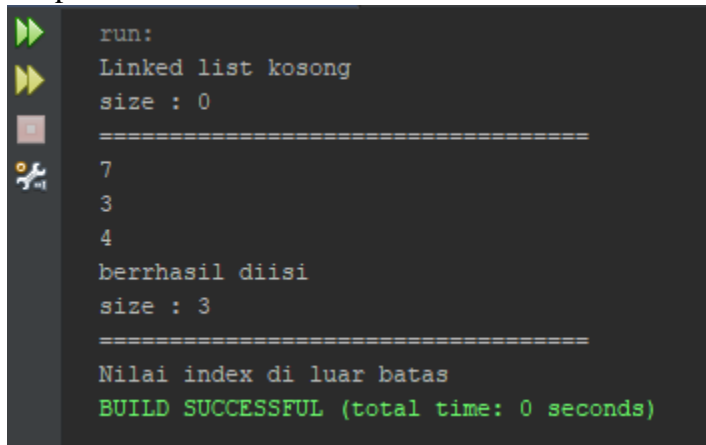
No	DoubleLinkedList.javaMain
1	package minggu11;
2	
3	public class doubleLinkedListMain {
4	
5	public static void main(String[] args) {
6	doubleLinkedList d = new doubleLinkedList();
7	try {
8	d.print();
9	System.out.println("size : " + d.size());
10	System.out.println("=====");
11	d.addFirst(3);
12	d.addLast(4);
13	d.addFirst(7);
14	d.print();
15	System.out.println("size : " + d.size());
16	System.out.println("=====");
17	d.add(40, 4);
18	d.print();
19	System.out.println("size : " + d.size());
20	System.out.println("=====");
21	d.addLast(50);
22	d.addLast(40);
23	d.addLast(10);
24	d.addLast(20);
25	d.print();
26	System.out.println("Size : " + d.size());
27	System.out.println("=====");
28	d.removeFirst();
29	d.print();
30	System.out.println("Size : " + d.size());

```

31      System.out.println("=====");
32      d.removeLast();
33      d.print();
34      System.out.println("Size : " + d.size());
35      System.out.println("=====");
36      d.remove(1);
37      d.print();
38      System.out.println("Size : " + d.size());
39      System.out.println("data pada awal linked list adalah : " + d.getFirst());
40      System.out.println("data pada awal akhir list adalah : " + d.getLast());
41      System.out.println("data yang di ambil dari method get : " + d.get(1));
42      d.print();
43  } catch (Exception e) {
44      System.out.println(e.getMessage());
45  }
46  }
47  }

```

Output :



```

run:
Linked list kosong
size : 0
=====
7
3
4
berrhasil diisi
size : 3
=====
Nilai index di luar batas
BUILD SUCCESSFUL (total time: 0 seconds)

```

## B. PERTANYAAN

1. Jelaskan pengertian dari double linked list!
2. Perhatikan class Node, di dalamnya terdapat atribut next dan prev.  
Untuk apakah atribut tersebut?
3. Perhatikan konstruktor pada class doubleLinkedList. Apa kegunaan inisialisasi atribut head dan size seperti gambar berikut?

```
11 public doubleLinkedList() {  
12     head = null;  
13     size = 0;  
14 }
```

4. Pada method addFirst(), kenapa dalam pembuatan object dari konstruktor class Node prev dianggap sama dengan null?  
Node newNode = new Node(null, item, head);
5. Perhatikan pada method addFirst(). Apakah arti statement head.prev = newNode ?
6. Perhatikan isi method addLast(), apa arti dari pembuatan object Node dengan mengisi parameter prev dengan current, dan next dengan null? Node newNode = new Node(current, item, null);
7. Apakah maksud statement berikut pada method removeFirst()?  
head = head.next;  
head.prev = null;
8. Bagaimana cara mendeteksi posisi data ada pada bagian akhir pada method removeLast()?
9. Apakah kegunaan method size() pada class doubleLinkedList?
10. Index pada linked list yang telah dibuat dalam praktikum, menunjukkan bahwa linked list dimulai ada indeks ke 0 atau 1? Jelaskan!
11. Jelaskan perbedaan fungsi Add pada double linkedlist dan Single Linkedlist! Kemudian tunjukkan perbedaannya pada kode program!
12. Jelaskan fungsi kode program berikut ini pada fungsi remove!

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

Jawab :

1. adalah linked list dengan node yang memiliki data dan dua buah reference link(biasanya disebut next dan prev) yang menunjuk ke node sebelum dan node sesudahnya.
2. Atribut next digunakan untuk : menunjuk pada node setelahnya dan pointer Atribut prev digunakan untuk : menunjuk pada node sebelumnya.
3. Inisialisasi atribut head digunakan untuk menunjukan Node pertama menunjuk ke null dan inisialisasi size digunakan untuk menunjuk ke pada Node indeks ke 0

4. Karena prev pada new Node akan menunjukan ke arah null maka konstruktor class node prev di anggap sama dengan null.
5. Statement ini berarti menunjukan bahwa arah panah prev dari head menunjuk ke node baru yang ditambahkan.
6. pada bagian akhir linked list diawali dengan penentuan Node akhir (last) sebagai lokasi yang akan ditambah, kemudian Node baru akan ditambahkan pada posisi lokasi next.
7. pada bagian awal dilakukan dengan pencarian lokasi awal double linked list, kemudian melakukan removing, dan menjadikan data pada bagian next menjadi bagian head.
8. Dengan cara menginisialisasi current.next = null atau bisa dengan kode program dibawah ini

```
Node current = head;
while(current.next.next != null){
    current = current.next;
}
current.next = null;
size--;
```

9. Digunakan untuk menunjukan berapa indeks yang terhubung ketika method size dipanggil pada fungsi main
10. Dimulai dari index ke 0 karena double linkedlist tetap menggunakan konsep array walaupun dinamis

#### 11. Single Linked List

```
public void add(int item, int index) throws Exception {
    if (index < 0 || index > size) throw new Exception("Nilai index diluar batas");
    if (isEmpty() || index == 0) {
        addFirst(item);
    }else{
        Node tmp=head;
        for (int i = 1; i < index; i++) {
            tmp = tmp.next;
        }
        Node next=(tmp == null) ? null :tmp.next;
        tmp.next=new Node(item, next);
        size++;
    }
}
```

Double Linked List

```
public void add(int item, int index) throws Exception{
    if (isEmpty()) addFirst(item);
    if (index < 0 || index > size) throw new Exception("Nilai index diluar 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);
            newNode.prev=current.prev;
            newNode.next=current;
        }
    }
}
```

12. Digunakan untuk melewati node sehingga node tersebut hilang.

### C. TUGAS

#### 1. Tugas1

No	doubleLinkedListT1
	<pre>package minggu11;  public class doubleLinkedListT1 {      Node head;     int size;      doubleLinkedListT1() {         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()) {</pre>



	<pre>addFirst(item);  } if (index &lt; 0    index &gt; size) {     throw new Exception("Nilai index di luar batas"); } else {     Node current = head;     int i = 0;     while (i &lt; 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;     } } size++; }  int size() {     return size; }  void clear() {     head = null;     size = 0; }  void print() {     if (!isEmpty()) {         Node tmp = head;         while (tmp != null) {             System.out.println(tmp.data + "\t");             tmp = tmp.next;         }         System.out.println("berhasil diisi");     } else {         System.out.println("Linked list kosong");     } }</pre>
--	---

```

    }
}

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--;
    }
}

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

void remove(int index) throws Exception {
    if (isEmpty() || index >= size) {
        throw new Exception("Nilai indeks di luar batas");
    }
    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--;
    }
}

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

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;
}

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

public void search(int cari) throws Exception {
    if (isEmpty()) {

```

	<pre>         throw new Exception("Nilai Indeks di Luar Batas !");     }     int index = 0;     int i = 0;     Node tmp = head;     while (tmp != null) {         if (cari == tmp.data) {             index = i;         }         tmp = tmp.next;         i++;     }     System.out.println("Angka yang Dicari Berada pada Indeks ke- " + index); } </pre>
--	---

No	doubleLinkedListMainT1
	<pre> package minggu11; import java.util.*; public class doubleLinkedListMainT1 {      static Scanner sc = new Scanner(System.in);      static void menu() {         System.out.println("DOUBLE LINK LIST DENGAN MENU");         System.out.println("=====");         System.out.println("Memlih menu");         System.out.println("1. Tambah awal");         System.out.println("2. Tambah Akhir");         System.out.println("3. Tambah Data Indeks Tertentu");         System.out.println("4. Hapus Awal");         System.out.println("5. Hapus AKhir");         System.out.println("6. Hapus Indeks Tertentu");         System.out.println("7. Cetak Data");         System.out.println("8. Cari");         System.out.println("9. Sorting");         System.out.println("10. Keluar");         System.out.println("=====");         System.out.print("&gt;&gt; ");     }      public static void main(String[] args) {         doubleLinkedListT1 dll = new doubleLinkedListT1();         int pil = 0;         try {             do { </pre>

```

menu();
pil = sc.nextInt();

System.out.println("=====");
switch (pil) {
    case 1: {
        System.out.print("Masukan data : ");
        dll.addFirst(sc.nextInt());

        break;

    }
    case 2: {
        System.out.print("Masukan data : ");
        int item = sc.nextInt();
        dll.addLast(item);

        break;
    }
    case 3: {
        System.out.print("Masukan data : ");
        int x = sc.nextInt();
        System.out.print("Masukan Posisi index data : ");
        int y = sc.nextInt();
        dll.add(x, y);
        break;
    }
    case 4: {
        dll.removeFirst();

        break;
    }
    case 5: {
        dll.removeLast();

        break;
    }
    case 6: {
        System.out.print("Masukan data yang ingin dihapus : ");
        dll.remove(sc.nextInt());
        break;
    }
    case 7: {
        dll.print();
        break;
    }
}

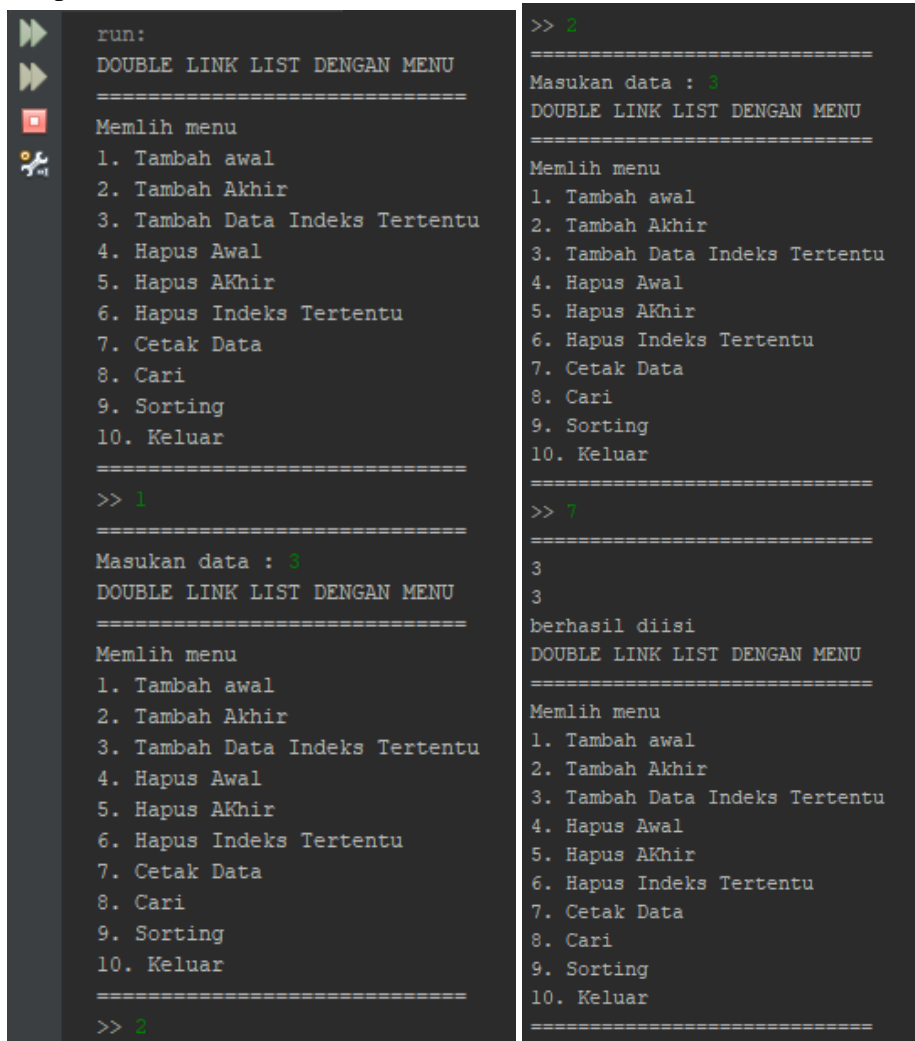
```

```

    }
    case 8: {
        System.out.print("Masukkan Data yang Ingin diCari = ");
        dll.search(sc.nextInt());
        break;
    }
}
} while (pil != 10);
} catch (Exception e) {
    System.out.println(e.getMessage());
}
}
}

```

Output :



```

run:
DOUBLE LINK LIST DENGAN MENU
=====
Memlih menu
1. Tambah awal
2. Tambah Akhir
3. Tambah Data Indeks Tertentu
4. Hapus Awal
5. Hapus AKhir
6. Hapus Indeks Tertentu
7. Cetak Data
8. Cari
9. Sorting
10. Keluar
=====
>> 1
=====
Masukan data : 3
DOUBLE LINK LIST DENGAN MENU
=====
Memlih menu
1. Tambah awal
2. Tambah Akhir
3. Tambah Data Indeks Tertentu
4. Hapus Awal
5. Hapus AKhir
6. Hapus Indeks Tertentu
7. Cetak Data
8. Cari
9. Sorting
10. Keluar
=====
>> 3

```

```

>> 3
=====
Masukan data : 3
DOUBLE LINK LIST DENGAN MENU
=====
Memlih menu
1. Tambah awal
2. Tambah Akhir
3. Tambah Data Indeks Tertentu
4. Hapus Awal
5. Hapus AKhir
6. Hapus Indeks Tertentu
7. Cetak Data
8. Cari
9. Sorting
10. Keluar
=====
>> 7
=====
3
3
berhasil diisi
DOUBLE LINK LIST DENGAN MENU
=====
Memlih menu
1. Tambah awal
2. Tambah Akhir
3. Tambah Data Indeks Tertentu
4. Hapus Awal
5. Hapus AKhir
6. Hapus Indeks Tertentu
7. Cetak Data
8. Cari
9. Sorting
10. Keluar
=====

```

## 2. Tugas2

No	NodeT2.java
	<pre> package minggu11;  public class NodeT2 {     int data;     Node prev, next;      NodeT2(Node prev, int data, Node next) {         this.prev = prev;         this.data = data;         this.next = next;     } } </pre>

No	DoubleLinkedList.java
	<pre> package minggu11;  public class doubleLinkedListT2 {      Node head;     int size;      doubleLinkedListT2() {         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);         }     } } </pre>

```

    } 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);
    }
    if (index < 0 || index > size) {
        throw new Exception("Nilai index 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);
            newNode.prev = current.prev;
            newNode.next = current;
            current.prev.next = newNode;
            current.prev = newNode;
        }
    }
    size++;
}

int size() {
    return size;
}

void clear() {
    head = null;
}

```



```

        size = 0;
    }

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

    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--;
        }
    }

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

```

```

void remove(int index) throws Exception {
    if (isEmpty() || index >= size) {
        throw new Exception("Nilai indeks di luar batas");
    }
    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--;
    }
}

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

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;
}

```

```

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

public void search(int cari) throws Exception {
    if (isEmpty()) {
        throw new Exception("Nilai Indeks di Luar Batas !");
    }
    int index = 0;
    int i = 0;
    Node tmp = head;
    while (tmp != null) {
        if (cari == tmp.data) {
            index = i;
        }
        tmp = tmp.next;
        i++;
    }
    System.out.println("Angka yang Dicari Berada pada Indeks ke- " +
index);

}

public void Sorting() {
    for (int i = 1; i < size; i++) {
        Node tmp = head;
        for (int j = 0; j < size - i; j++) {
            if (tmp.data > tmp.next.data) {
                int simpan = tmp.data;
                tmp.data = tmp.next.data;
                tmp.next.data = simpan;
            }
            tmp = tmp.next;
        }
    }
}
}

```

