9.3.5. Insert the Node at given position in D.L.L

Lab16.java

```
package com.jlcindia.linkedlist.doubly;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Lab16 {

static Node insert(Node headNode, int position, int data) {

		//1. Create New Node
		Node temp = new Node(data);

		//2.when head is null
		if(headNode==null) {
			return temp;
		}

		//3. If position is 1
		if(position==1){
			temp.next = headNode;
			headNode.prev=temp;
			return temp;
		}

		//4. Move the position -1 Node
		Node currentNode = headNode;
		for(int i=1;i<=position-2 && currentNode != null;i++) {
			currentNode=currentNode.next;
		}

		//5. If Position is out of range
		if(currentNode==null) {
			return headNode;
		}
```

```
            //6. Insert at given Position
            temp.next=currentNode.next;
            temp.prev=currentNode;

            currentNode.next.prev=temp;
            currentNode.next = temp;

            return headNode;

        }

        static Node insertLast(Node headNode,int data) {
            // Copy from Lab15
        }

        static String toString(Node headNode) {
            // Copy from Lab15
        }

        public static void main(String[] args) {

        Node head=null;
        head = insert(head,3,55);
        head = insertLast(head,10);
        head = insertLast(head,20);
        head = insertLast(head,30);
        head = insertLast(head,40);
        head = insertLast(head,50);

        System.out.println(toString(head));
        head = insert(head,3,99); //
        System.out.println(toString(head));
        }
}
```

9.3.6. Delete the First Node in D.L.L

Lab17.java

```
package com.jlcindia.linkedlist.doubly;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Lab17 {

	static Node deleteFirst(Node headNode) {

		//1. If head is null
		if(headNode==null) {
				return null;
		}

		//2. If One Node is there
		if(headNode.next==null){
			return null;
		}

		//3.If Head is Not Null
		Node temp = headNode;

		headNode = headNode.next;
		headNode.prev=null;

		temp.next = null;

		return headNode;
	}

	static Node insertLast(Node headNode,int data) {
		 // Copy from Lab15
	}
```

```
        static String toString(Node headNode) {
                // Copy from Lab15
        }

        public static void main(String[] args) {

        Node head=null;
        head = insertLast(head,10);
        head = insertLast(head,20);
        head = insertLast(head,30);
        head = insertLast(head,40);
        head = insertLast(head,50);

        System.out.println(toString(head));
        head = deleteFirst(head); //10
        System.out.println(toString(head));
        head = deleteFirst(head); //20
        System.out.println(toString(head));

        }
}
```

9.3.7. Delete the Last Node in D.L.L

Lab18.java

```
package com.jlcindia.linkedlist.doubly;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Lab18 {

	static Node deleteLast(Node headNode) {

		//1. If head is null
		if(headNode==null) {
				return null;
		}

		//2. If One Node is there
		if(headNode.next==null){
			return null;
		}

		//3. Move to Last but one
		Node currentNode = headNode;
		while (currentNode.next.next != null) {
			currentNode = currentNode.next;
		}

		Node temp = currentNode.next;
		temp.prev=null;

		currentNode.next = null;
		return headNode;
	}
```

```
        static Node insertLast(Node headNode,int data) {
                // Copy from Lab15
        }


         static String toString(Node headNode) {
                // Copy from Lab15
        }

        public static void main(String[] args) {

        Node head=null;
        head = insertLast(head,10);
        head = insertLast(head,20);
        head = insertLast(head,30);
        head = insertLast(head,40);
        head = insertLast(head,50);

        System.out.println(toString(head));
        head = deleteLast(head); //50
        System.out.println(toString(head));
        head = deleteLast(head); //40
        System.out.println(toString(head));

        }

}
```

9.3.8. Delete the Node at given position in D.L.L

Lab19.java

```
package com.jlcindia.linkedlist.doubly;

/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

public class Lab19 {

	static Node delete(Node headNode, int position) {

		// 1.head is null
		if (headNode == null)
			return null;

		// 2.One Node
		if (headNode.next == null)
			return null;

		// 3.delete first node
		if (position == 1) {
			Node temp = headNode;

			headNode = headNode.next;
			headNode.prev = null;

			temp.next = null;
			return headNode;
		}
```

```
        // 4. Move to Position -1
        Node currentCode = headNode;

        for (int i = 1; i <= position - 2 && currentCode != null; i++) {
                currentCode = currentCode.next;
        }

        // 5.Position is out of Range
        if (currentCode == null || currentCode.next==null) {
                return headNode;
        }

        Node temp = currentCode.next;

        currentCode.next = currentCode.next.next;
        if (currentCode.next != null) {
                currentCode.next.prev = currentCode;
        }

         temp.next = null;
         temp.prev = null;

        return headNode;
}

static Node insertLast(Node headNode,int data) {
         // Copy from Lab15
}


 static String toString(Node headNode) {
         // Copy from Lab15
}
```

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

                Node head = null;
                head = insertLast(head, 10);
                head = insertLast(head, 20);
                head = insertLast(head, 30);
                head = insertLast(head, 40);
                head = insertLast(head, 50);

                System.out.println(toString(head));
                head = delete(head, 3); // 30
                System.out.println(toString(head));
                head = delete(head, 4); // 50
                System.out.println(toString(head));
                head = delete(head, 4); // No Deletion
                System.out.println(toString(head));
                head = delete(head, 3); // 40
                System.out.println(toString(head));
        }

}
```

9.3.9. Search the Node in D.L.L from beginnig

Lab20.java

```
package com.jlcindia.linkedlist.doubly;

/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Lab20{

        static int search(Node headNode, int element) {

                int position =1;

                Node currentNode = headNode;

                while(currentNode!=null) {
                        if( currentNode.data == element) {
                                return position;
                        }else {
                                position++;
                                currentNode = currentNode.next;
                        }
                }
                return -1;
        }

        static Node insertLast(Node headNode,int data) {
                // Copy from Lab15
        }

        static String toString(Node headNode) {
                // Copy from Lab15
        }
```

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

                Node head = null;
                head = insertLast(head, 10);
                head = insertLast(head, 20);
                head = insertLast(head, 30);
                head = insertLast(head, 40);
                head = insertLast(head, 50);

                System.out.println(toString(head));
                int x = search(head, 30);  //3
                System.out.println(x);
                x = search(head, 50);  //5
                System.out.println(x);
                x = search(head, 60);  //-1
                System.out.println(x);
        }

}
```

9.3.10. Search the Node in D.L.L from ending

Lab21.java

```
package com.jlcindia.linkedlist.doubly;

/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

public class Lab21{

	static int search(Node headNode, int element) {

		Node currentNode = headNode;
		int nodeCount = 1;

		//Move to Last Node
		while(currentNode.next != null) {
			currentNode = currentNode.next;
			nodeCount++;
		}

		while(currentNode!=null) {
			if( currentNode.data == element) {
				return nodeCount;
			}else {
				nodeCount--;
				currentNode = currentNode.prev;
			}
		}

		return -1;
	}
```

```
    static Node insertLast(Node headNode,int data) {
            // Copy from Lab15
    }


     static String toString(Node headNode) {
            // Copy from Lab15
    }

    public static void main(String[] args) {

            Node head = null;
            head = insertLast(head, 10);
            head = insertLast(head, 20);
            head = insertLast(head, 30);
            head = insertLast(head, 30);
            head = insertLast(head, 30);
            head = insertLast(head, 40);
            head = insertLast(head, 50);

            System.out.println(toString(head));
            int x = search(head, 30);  //3
            System.out.println(x);
            x = search(head, 50);  //5
            System.out.println(x);
            x = search(head, 70);  //-1
            System.out.println(x);
    }

}
```

9.3.11. Reverse the D.L.L

Lab22.java

```
package com.jlcindia.linkedlist.doubly;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Lab22 {
	static Node reverse(Node headNode) {

		// 1. Empty List
		if (headNode == null) {
			return null;
		}
		// 2. One Node List
		if (headNode.next == null) {
			return headNode;
		}

		Node tempNode = null;

		Node currentNode = headNode;
		while(currentNode!=null) {
			//Swap
			tempNode = currentNode.prev;
			currentNode.prev= currentNode.next;
			currentNode.next = tempNode;

			currentNode = currentNode.prev;
		}

		headNode = tempNode.prev;
		return headNode;
	}
```

```
        static Node insertLast(Node headNode,int data) {
                // Copy from Lab15
        }

        static String toString(Node headNode) {
                // Copy from Lab15
        }

        public static void main(String[] args) {

                Node head = null;
                head = insertLast(head, 10);
                head = insertLast(head, 20);
                head = insertLast(head, 30);
                head = insertLast(head, 40);
                head = insertLast(head, 50);

                System.out.println(toString(head));
                head = reverse(head);
                System.out.println(toString(head));

        }

}
```

9.3.12. Design MyDoublyLinkedList

Lab23.java

```
package com.jlcindia.linkedlist.doubly;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

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

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

public class MyDoublyLinkedList {

	Node headNode = null;
	int nodeCount = 0;

	public MyDoublyLinkedList() {
	}

	public String toString() {
		if (this.headNode == null) {
			return "[]";
		}

		String str = "[";
		Node currentNode = this.headNode;
```

```
        while (currentNode != null) {
                str = str + "" + currentNode.data + " , ";
                currentNode = currentNode.next;
        }
        str = str.substring(0, str.length() - 2);
        str = str + "]";

        return str;
}

public boolean isEmpty() {
        return this.nodeCount == 0;
}

public int size() {
        return this.nodeCount;
}

public void clear() {

        Node currentNode = this.headNode;
        while (currentNode != null) {
                Node temp = currentNode.next;

                currentNode.next = null;
                currentNode.prev = null;

                currentNode = temp;
        }

        this.headNode = null;
        this.nodeCount = 0;
}
```

```
public void insertFirst(int data) {

        Node temp = new Node(data);

        // 1.Empty List
        if (this.headNode == null) {
                this.headNode = temp;
                nodeCount++;
                return;
        }

        // 2.List with 1 or more Nodes
        temp.next = this.headNode;
        this.headNode.prev = temp;

        this.headNode = temp;
        nodeCount++;
}

public void insertLast(int data) {

        Node temp = new Node(data);

        // 1.Empty List
        if (this.headNode == null) {
                this.headNode = temp;
                nodeCount++;
                return;
        }

        // 2.List with 1 or more Nodes
        Node currentNode = this.headNode;
        while (currentNode.next != null) {
                currentNode = currentNode.next;
        }
```

```
        currentNode.next = temp;
        temp.prev = currentNode;
        nodeCount++;
}

public void insert(int position, int data) {

        // 1. Create New Node
        Node temp = new Node(data);

        // 2.Empty List
        if (this.headNode == null) {
                this.headNode = temp;
                nodeCount++;
                return;
        }

        // 3. If position is 1
        if (position == 1) {
                temp.next = this.headNode;
                this.headNode.prev = temp;
                this.headNode = temp;
                nodeCount++;
                return;
        }

        // 4. Move the position -1 Node
        Node currentNode = this.headNode;
        for (int i = 1; i <= position - 2 && currentNode != null; i++) {
                currentNode = currentNode.next;
        }

        // 5. If Position is out of range
        if (currentNode == null) {
                return;
        }
```

```
        // 6. Insert at given Position
        temp.next = currentNode.next;
        temp.prev = currentNode;

        currentNode.next.prev = temp;
        currentNode.next = temp;

        nodeCount++;

}

public void deleteFirst() {

        // 1. If head is null
        if (this.headNode == null) {
                this.headNode = null;
                return;
        }

        // 2. If One Node is there
        if (this.headNode.next == null) {
                this.headNode = null;
                nodeCount--;
                return;
        }

        // 3.when more nodes are there
        Node temp = this.headNode;

        this.headNode = this.headNode.next;
        headNode.prev = null;

        temp.next = null;
        nodeCount--;

}
```

```
public void deleteLast() {

        // 1. If head is null
        if (this.headNode == null) {
                return;
        }

        // 2. If One Node is there
        if (this.headNode.next == null) {
                this.headNode = null;
                nodeCount--;
                return;
        }

        // 3. When two or More Nodes
        Node currentNode = this.headNode;
        while (currentNode.next.next != null) {
                currentNode = currentNode.next;
        }

        Node temp = currentNode.next;
        temp.prev = null;

        currentNode.next = null;
        nodeCount--;

}

public void delete(int position) {

        // 1.head is null
        if (this.headNode == null) {
                this.headNode = null;
                return;
        }
```

```
// 2.One Node
if (this.headNode.next == null) {
        this.headNode = null;
        nodeCount--;
        return;
}

// 3.delete first node
if (position == 1) {
        Node temp = this.headNode;

        this.headNode = this.headNode.next;
        this.headNode.prev = null;

        temp.next = null;
        nodeCount--;

        return;
}

// 4. Move to Position -1
Node currentCode = this.headNode;

for (int i = 1; i <= position - 2 && currentCode != null; i++) {
        currentCode = currentCode.next;
}

// 5.Position is out of Range
if (currentCode == null || currentCode.next == null) {
        return;
}

Node temp = currentCode.next;

currentCode.next = currentCode.next.next;
if (currentCode.next != null) {
        currentCode.next.prev = currentCode;
```

```
        }

        temp.next = null;
        temp.prev = null;

        nodeCount--;

    }

    public int searchFirst(int element) {

        int position = 1;
        Node currentNode = headNode;

        while (currentNode != null) {
            if (currentNode.data == element) {
                return position;
            } else {
                position++;
                currentNode = currentNode.next;
            }
        }
        return -1;
    }

    public int searchLast(int element) {

        Node currentNode = headNode;
        int nodeCount = 1;

        // Move to Last Node
        while (currentNode.next != null) {
            currentNode = currentNode.next;
            nodeCount++;
        }
```

```
            while (currentNode != null) {
                if (currentNode.data == element) {
                    return nodeCount;
                } else {
                    nodeCount--;
                    currentNode = currentNode.prev;
                }
            }
            return -1;
        }

        public void reverse() {
            // 1. Empty List
            if (this.headNode == null) {
                return;
            }
            // 2. One Node List
            if (this.headNode.next == null) {
                return;
            }

            Node tempNode = null;

            Node currentNode = this.headNode;
            while (currentNode != null) {
                // Swap
                tempNode = currentNode.prev;
                currentNode.prev = currentNode.next;
                currentNode.next = tempNode;

                currentNode = currentNode.prev;

            }
            this.headNode = tempNode.prev;
        }
}
```

```
public class Lab23 {

	public static void main(String[] args) {

		MyDoublyLinkedList mylist = new MyDoublyLinkedList();

		System.out.println("---------1-----------");
		System.out.println(mylist);
		System.out.println(mylist.isEmpty());
		System.out.println(mylist.size());

		mylist.insertFirst(10);
		mylist.insertFirst(20);
		mylist.insertFirst(30);

		System.out.println("---------2-----------");
		System.out.println(mylist);
		System.out.println(mylist.isEmpty());
		System.out.println(mylist.size());

		mylist.insertLast(88);
		mylist.insertLast(99);

		System.out.println("---------3-----------");
		System.out.println(mylist);
		System.out.println(mylist.isEmpty());
		System.out.println(mylist.size());


		mylist.insert(4, 77);

		System.out.println("---------4-----------");
		System.out.println(mylist);
```

```
                mylist.deleteFirst();

                System.out.println("---------5----------");
                System.out.println(mylist);

                mylist.deleteLast();

                System.out.println("---------6----------");
                System.out.println(mylist);

                mylist.delete(2);

                System.out.println("---------7----------");
                System.out.println(mylist);

                mylist.insertLast(77);
                mylist.insertLast(99);

                System.out.println("---------8----------");
                System.out.println(mylist);

                System.out.println("---------9----------");
                System.out.println(mylist.searchFirst(77));
                System.out.println(mylist.searchLast(77));
                System.out.println(mylist.searchFirst(55));
                System.out.println(mylist.searchLast(55));

                mylist.reverse();
                System.out.println("---------10----------");
                System.out.println(mylist);
        }
}
```