# Worksheet-5

**Student Name: Arpit Anand**

**UID: 23BCS12710**
**Branch: BE - CSE**
**Section/Group:KRG-3(A)**
**Semester: 5**
**Subject Name: DAA**

**Date of Performance: 12/08/2025**
**Subject Code: 23CSH-301**

**1.Aim:** Apply the concept of Linkedlist and write code to Insert and Delete an element at the beginning and at end in a Circular Linked List.

**2.Objective:** To understand and apply the concepts of Linked List (Circular) in Java by implementing insertion and deletion operations at both the beginning and end of it. This exercise reinforces knowledge of dynamic data structures, pointer manipulation, and efficient node operations in different linked list implementations.

**3.Requirements(Hardware/Software):**Online Java compiler.

## 4.Algorithm:

1. Create Node→Stored at a and a pointer to the next node.
2. Insertion
   - If list isempty →point new node to itself.
   - Else→insert at desired position and adjust pointers to maintain circular link.
3. Deletion
   - Find node to delete.
   - Adjust previous node's pointer to skip deleted node.
   - If deleting last node→ update last pointer.
4. Traversal
   - Start from head, move to next node until your each head again.
5. Display.
6. End.

## 5.Procedure:

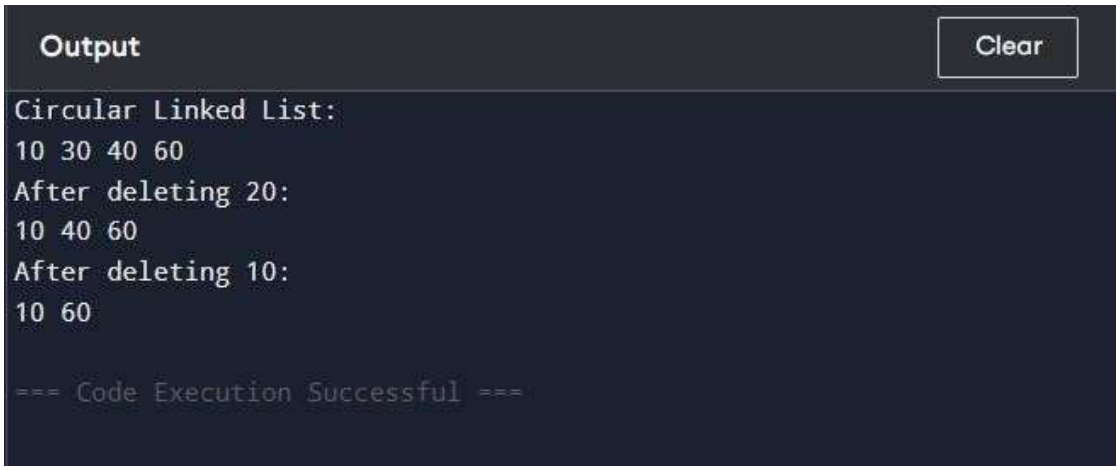
```
public class Main {
static class Node {
intdata;Nodenext;
Node(int data) {
this.data = data;
this.next = null;
} }
staticclassCircularLinkedList{ Node
head = null;
Nodetail= null;
voidinsert(intdata){
NodenewNode=newNode(data); if
(head == null) {
head=newNode;
tail=newNode; tail.next= head;
}else{
tail.next=newNode;
tail = newNode;
tail.next = head;
} }
voiddelete(intkey){ if
(head == null) {
System.out.println("Listisempty.");
return; }
Nodecurrent=head,previous=null; if
(current.data == key) {
if(head==tail){
head=null;
tail = null; } else {
head = head.next;
tail.next = head;
previous = current;
current=current.next;
if (current.data == key) {
previous.next=current.next; if
(current == tail) {
tail=previous;
}
return;
}
}while(current!=head);
System.out.println("Valuenotfound.");}voiddisplay(){ if
(head == null) {
System.out.println("Listisempty.");
return;
}
Nodecurrent=head; do
{
```

```
System.out.print(current.data+"");
current = current.next;
}
while(current!=head);
System.out.println();
} }
public static void main(String[] args) {
CircularLinkedListcll=newCircularLinkedList();
cll.insert(10);
cll.insert(30);
cll.insert(40);
cll.insert(60);
System.out.println("CircularLinkedList:");
cll.display();
cll.delete(30);
System.out.println("Afterdeleting20:");
cll.display();
cll.delete(40);
System.out.println("Afterdeleting10:");
cll.display();
}
```

**Time Complexity : O(n)**

**SpaceComplexity: O(n)**

**Output:**




**Learning Outcomes:**

1. Understood circular linked list usage.
2. Deeper understanding linkedlists and their creation.
3. Learned about logic of linkedlist usage in real world usage.