# TRIBHUWAN UNIVERSITY JANAPRIYA MULTIPLE CAMPUS

Simalchaur, Pokhara-8, Nepal



## LAB REPORT

Data Structure and Algorithm with JAVA

# Submitted To:

Mr.Prithvi Raj Paneru

Data Structure and Algorithm with JAVA

Submitted by: Ms. Krishna Luharuka

4<sup>th</sup> Semester

Bachelor in Information Management

Roll No: 11

Mangsir,2079

# **INDEX:**

SN	Topic	Date of	Date of	Signature	Remark
		Experiment	submission		
1.	Operation in a singly linked list.	11-11-2022	24 -11-2022		
2.	Operation in a doubly linked list.	24-11-2022			
3	Operation in a circular singly linked list.	01-12-2022			
4	Operation in a circular doubly linked list.				

## **EXPERIMENT: 1**

## TITLE:

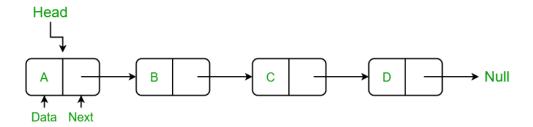
A java program to run various operations of singly linked list.

## 1.OBJECTIVE:

- ✓ To insert a node at beginning, at last and at certain position and display it.
- ✓ To delete a node from the beginning, from last and from certain position.
- ✓ To search a node from singly linked list.

## 2.THEORY:

Singly linked list is the data structure used for storing collection of nodes and the node has data and reference to next node in a list. The first node is the head node and the last node has data and points to null.



The various operations of singly linked list are discussed below:

#### 2.1. Traverse:

The traverse operation helps to display the content of a singly linked list. Here, to run this operation, the temp node is kept moving to the next one and the content is displayed.

## 2.2.Insertion:

The insertion operation inserts a node in linked list. The insertion operation in singly linked list can be run in three different methods. They are:

## **2.2.1.insertatbeg(int x)**:

In this method, a node is inserted at the beginning of the singly linked list.

## 2.2.2.insertatlast(int x):

In this method, a node is inserted at the last of the linked list.

## 2.2.3.insertatpos(int x, int pos):

This method inserts a node at the specified position of the linked list.

#### 2.3. Deletion:

The deletion operation deletes a node from the linked list. The deletion operation in singly linked list can also be run in three different methods. They are:

## **2.3.1.deleteatbeg()**:

In this method, a node is deleted from the very beginning of the singly linked list.

## 2.3.2.deleteatlast():

In this method, a node is deleted from the last of the singly linked list.

## 2.3.3.deleteatpos(int pos):

In this method, a node is deleted from the specified position of the linked list.

#### 2.4. Search:

The search(int key) operation helps to search a key from the nodes of the linked list and return true if the key exists in the node and return false if the key doesn't exist in the singly list.

## **3.IMPLEMENTATION:**

The following programs shows the insertion of nodes at the beginning, at last and at certain position in singly linked list data structure in java. Similarly, the deletion of node from the beginning, from the last and from the specified position is also shown. Moreover, the source code and output for search operation is also shown.

# 3.1.PROGRAM 1: Traverse a singly linked list.

## **PSEUDOCODE:**

```
public void display()
    {
        node temp=head;
        while(temp!=null)
        {
            System.out.print(temp.data+"-->");
            temp=temp.next;
        }
        System.out.print(temp+"\n");
     }
```

## **SOURCE CODE:**

```
class SllForTraverse{
                                                    System.out.print("\n");
  private static class node {
    int data;
    node next;
                                                  public static void main(String args[])
    node(int data) {
       this.data=data;
                                                    SllForTraverse s=new
       this.next=null;
                                               SllForTraverse();
                                                    s.head = new node(11);
                                                    node second=new node(12);
  public node head=null;
                                                    node third=new node(13);
                                                    node fourth=new node(14);
                                                    s.head.next=second;
  public void display(){
    node temp=head;
                                                    second.next=third;
    while(temp!=null){
                                                    third.next=fourth;
       System.out.print(temp.data+"-->");
                                                    System.out.println("A singly linked
       temp=temp.next;
                                               list traversed is given as:");
                                                    s.display();
    System.out.print(temp+"\n");
                                                  }}
System.out.println("*************
```

**3.2.PROGRAM 2**: Insert Element to a Linked List either at the beginning, middle or end of the linked list.

## 3.2.1. Insert at the beginning:

#### **PSEUDOCODE:**

```
public void insertatbeg(int x)
{
    node newnode=new node(x);
    if(head==null)
    {
        head=newnode;
    }
    else
    {
        newnode.next=head;
        head=newnode;
    }
}
```

## 3.2.2. Insert at the last:

#### PSEUDOCODE:

```
public void insertatlast(int x)
{
    node newnode=new node(x);
    if(head==null)
    {
        head=newnode;
    }
    else
    {
        node temp=head;
        while(temp.next!=null)
        {
            temp=temp.next;
        }
        temp.next=newnode;
    }
}
```

## 3.2.3. Insert at specified position:

#### PSEUDOCODE:

```
public void insertatpos(int x,int pos)
                                                       else
    node\ newnode=new\ node(x);
                                                         node temp=head;
                                                         for(int i=1; i < pos-
    if(pos<1)
                                                  1\&\&temp!=null;i++)
       System.out.println("position less
than 1 is invalid");
                                                           temp=temp.next;
    else\ if(pos==1)
                                                         if(temp!=null)
                                                         newnode.next=temp.next;
       if(head = = null)
                                                         temp.next=newnode;
         head=newnode;
                                                         else
       else
                                                           System.out.println("invalid
         newnode.next=head;
                                                 position for insert");
         head=newnode;
```

#### **SOURCE CODE OF INSERTION OPERATION:**

```
import java.util.Scanner;
class SLL
                                                      node\ newnode=new\ node(x);
                                                      if(pos<1)
  class node
                                                        System.out.println("position less
    int data:
                                                 than 1 is invalid");
    node next;
    node(int data)
                                                      else\ if(pos==1)
       this.data=data;
                                                        if(head = = null)
       this.next=null;
                                                           head=newnode;
  public node head=null;
                                                        else
  public void insertatbeg(int x)
                                                           newnode.next=head;
                                                           head=newnode;
    node\ newnode=new\ node(x);
    if(head = = null)
                                                      else
       head=newnode;
                                                        node temp=head;
    else
                                                        for(int i=1; i < pos-
                                                 1&&temp!=null;i++)
       newnode.next=head;
       head=newnode;
                                                           temp=temp.next;
                                                        if(temp!=null)
  public void insertatlast(int x)
                                                        newnode.next=temp.next;
                                                        temp.next=newnode;
    node\ newnode=new\ node(x);
    if(head = = null)
                                                        else
       head=newnode;
                                                           System.out.println("invalid
                                                 position for insert");
    else
       node temp=head;
       while(temp.next!=null)
                                                   public void display()
         temp=temp.next;
                                                      node temp=head;
                                                      while(temp!=null)
       temp.next=newnode;
                                                        System.out.print(temp.data+"-->");
                                                        temp=temp.next;
  public void insertatpos(int x,int pos)
```

```
System.out.print(temp+"\n");
                                                            break;
System.out.println("************
case 2:
    System.out.print("\n");
                                                            s.insertatlast(11);
                                                            s.insertatlast(15);
                                                            s.display();
  public static void main(String args[])
                                                           break;
                                                         }
     SLL s=new SLL();
                                                         case 3:
     System.out.println("press 1 for
insertion at beginning");
                                                            s.insertatpos(1, -1);
    System.out.println("press 2 for
                                                            s.insertatpos(1,1);
insertion at last");
                                                            s.insertatpos(3,2);
     System.out.println("press 3 for
                                                            s.insertatpos(6,2);
insertion at position");
                                                            s.insertatpos(10,10);
     System.out.println("enter your
                                                            s.display();
choice");
                                                            break;
    Scanner input=new
Scanner(System.in);
                                                         default:
     int ch=input.nextInt();
    switch(ch)
                                                            System.out.println("please enter
                                                  your choice properly");
       case 1:
          s.insertatbeg(9);
          s.insertatbeg(10);
          s.display();
```

```
press 1 for insertion at beginning
press 2 for insertion at last
press 3 for insertion at position
enter your choice
10-->9-->null
**********
press 1 for insertion at beginning
press 2 for insertion at last
press 3 for insertion at position
enter your choice
2
11-->15-->null
**********
press 1 for insertion at beginning
press 2 for insertion at last
press 3 for insertion at position
enter your choice
position less than 1 is invalid
invalid position for insert
1-->6-->3-->null
**********
press 1 for insertion at beginning
press 2 for insertion at last
press 3 for insertion at position
enter your choice
4
please enter your choice properly
```

**3.3.PROGRAM 3**: Delete Element from a Linked List either from the beginning, middle or end of the linked list.

## **3.3.1. Deletion from the beginning:**

### PSEUDOCODE:

```
public void deleteatbeg()
    {
        if(head==null)
        {
            System.out.println("Empty list");
        }
        else if(head.next==null)
        {
            head=null;
        }
        else
        {
            head=head.next;
        }
    }
}
```

## 3.3.2. Deletion from the last:

## **PSEUDOCODE:**

```
public void deleteatlast()

{
    if(head==null)
    {
        System.out.print("empty list");
    }
    else if(head.next==null)
    {
        head=null;
    }
    else
    {
        node temp=head;
        while(temp.next.next!=null)
        {
            temp=temp.next;
        }
        temp.next=null;
    }
}
```

## 3.3.3. Deletion from the specified position:

## PSEUDOCODE:

```
public void deleteatpos(int pos){
                                                   else
if(pos<1){
  System.out.println("pos invalid");
                                                     node temp=head;
                                                    for(int i=1;i<pos-
else if(pos==1){
                                                1&&temp.next!=null;i++)
  if(head==null)
                                                      temp=temp.next;
    System.out.print("empty list");
                                                     if(temp.next!=null)
  else if(head.next==null) {
    head=null;
                                                      temp.next=temp.next.next;
  }
                                                     }else{
                                                      System.out.println("invalid pos");
  else {
    head=head.next;
```

#### **SOURCE CODE OF DELETE OPERATION:**

```
import java.util.Scanner;
                                                         while(temp.next.next!=null)
class SLLfordelete
                                                           temp=temp.next;
  private static class node
                                                         temp.next=null;
    int data;
    node next;
    node(int data)
                                                    public void deleteatpos(int pos)
      this.data=data;
                                                      if(pos<1)
      this.next=null;
                                                       System.out.println("pos invalid");
  public node head=null;
                                                      else if(pos==1)
  public void deleteatbeg()
                                                       if(head==null)
    if(head==null)
                                                         System.out.print("empty list");
      System.out.println("Empty list");
                                                       else if(head.next==null)
    else if(head.next==null)
                                                         head=null;
      head=null;
                                                       else
    else
                                                         head=head.next;
      head=head.next;
                                                     else
                                                       node temp=head;
  public void deleteatlast()
                                                       for(int i=1;i<pos-
                                                  1&&temp.next!=null;i++)
    if(head==null)
                                                         temp=temp.next;
      System.out.print("empty list");
                                                       if(temp.next!=null)
    else if(head.next==null)
                                                         temp.next=temp.next.next;
      head=null;
    }
    else
                                                         System.out.println("invalid pos");
      node temp=head;
```

```
}
                                                       System.out.print("enter your choice
                                                  \t");
  public void display()
                                                       Scanner input=new
                                                  Scanner(System.in);
    node temp=head;
                                                       int ch=input.nextInt();
    while(temp!=null)
                                                       switch(ch)
      System.out.print(temp.data+"-->");
                                                         case 1:
       temp=temp.next;
                                                           s.deleteatbeg();
    System.out.print(temp+"\n");
                                                           s.display();
                                                           break;
System.out.println("************
                                                         }
************");
                                                         case 2:
    System.out.print("\n");
  }
                                                           s.deleteatlast();
                                                           s.display();
  public static void main(String args[])
                                                           break;
    SLLfordelete s=new SLLfordelete();
                                                         case 3:
    s.head= new node(11);
    node second=new node(12);
                                                           s.deleteatpos(-1);
    node third=new node(13);
                                                           s.deleteatpos(1);
    node fourth=new node(14);
                                                           s.deleteatpos(2);
    s.head.next=second;
                                                           s.deleteatpos(10);
    second.next=third;
                                                           s.display();
    third.next=fourth;
                                                           break;
    System.out.println("Given a singly
linked list:");
                                                         default:
    s.display();
    System.out.println("press 1 for
                                                           System.out.println("please enter
deletion from the beginning");
                                                  your choice properly");
    System.out.println("press 2 for
deletion from the last");
                                                      }
    System.out.println("press 3 for delete
from the certain position");
```

```
Given a singly linked list:
11-->12-->13-->14-->null
press 1 for deletion from the beginning
press 2 for deletion from the last
press 3 for delete from the certain position
                    1
enter your choice
12-->13-->14-->null
**********
Given a singly linked list:
11-->12-->13-->14-->null
press 1 for deletion from the beginning
press 2 for deletion from the last
press 3 for delete from the certain position
enter your choice
                    2
11-->12-->13-->null
**********
Given a singly linked list:
11-->12-->13-->14-->null
press 1 for deletion from the beginning
press 2 for deletion from the last
press 3 for delete from the certain position
enter your choice
pos invalid
invalid pos
12-->14-->null
**********
Given a singly linked list:
11-->12-->13-->14-->null
press 1 for deletion from the beginning
press 2 for deletion from the last
press 3 for delete from the certain position
enter your choice
please enter your choice properly
```

#### **3.4.PROGRAM 4**: Search an element from a Linked List.

## **PSEUDOCODE:**

```
public boolean search(int key)
    {
        node temp=head;
        while(temp!=null)
        {
            if(temp.data==key)
            {
                return true;
            }
            temp=temp.next;
        }
        return false;
    }
}
```

#### **SOURCE CODE:**

```
class SllForSearch
                                                     node temp=head;
                                                     while(temp!=null)
  private static class node
    int data;
                                                       System.out.print(temp.data+"-->");
    node next;
                                                       temp=temp.next;
    node(int data)
                                                     System.out.print(temp+"\n");
       this.data=data;
       this.next=null;
                                                System.out.println("*************
                                                System.out.print("\n");
  public node head=null;
  public boolean search(int key)
                                                  public static void main(String args[])
    node temp=head;
                                                     SllForSearch s=new SllForSearch();
    while(temp!=null)
                                                     s.head = new node(11);
                                                     node\ second=new\ node(12);
       if(temp.data = = key)
                                                     node third=new node(13);
                                                     node fourth=new node(14);
                                                     s.head.next=second;
         return true;
                                                     second.next=third;
       temp=temp.next;
                                                     third.next=fourth;
                                                     System.out.println("Given singly
                                                linked list:");
    return false;
                                                     s.display();
                                                     System.out.println("Search 11");
  public void display()
```

```
Given singly linked list:

11-->12-->13-->14-->null

*********************

Search 11

element found

Search 1

Element not found
```

#### 4.OUTPUT AND DISCUSSION:

The output of program 1 shows that when the traverse() method is run the elements of the singly list are displayed in the output screen.

The output of the Program 2 shows that when the insertatbeg(x) method is run the entered values are inserted at the beginning in the singly linked list. Similarly, when the insertatlast(x) method is run, the head is null so the entered

node automatically becomes the last and the following nodes entered inserts at the end in singly linked list manner. Finally, when the insertatpos(x,pos) is run, pos<1or more than the size of nodes in singly linked list it displays invalid result. Furthermore, it also shows that if any valid or specific position is entered, it inserts the node at the specific position in the singly linked list.

In the output of Program 3, it can be seen that when the user choose choice 1, it takes to deleteatbeg() method and deletes the first node of the linked list.

Similarly, when choose 2, deleteatlast() is run which deletes the last node of the list. Besides, when choose 3, deleteatpos(pos) method is run and checking various conditions it delete a node from the specified position. Finally, if any number other than 1,2,3 is entered, a message to enter the proper choice is displayed.

Finally, in the output of Program 4, it is seen that any node can be searched from a given linked list by traversing to each of the node and returning true if the desired key node is found.

Therefore, The output of program 1, 2 and 3 supports the traverse of a nodes in a linked list, insertion of nodes, deletion of nodes and searching of a node from the singly linked list.

## **5.CONCLUSION:**

A Java program to traverse, insert a node at first, last, certain position, delete a node at beginning, at last and at certain position and search a node from the singly linked list is successfully run.

## **EXPERIMENT: 2**

## TITLE:

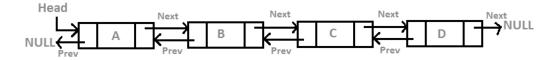
A java program to run various operations of doubly linked list.

## 1.OBJECTIVE:

- ✓ To insert a node at beginning, at last and at certain position and display it.
- ✓ To delete a node from the beginning, from last and from certain position.
- ✓ To sort the nodes from doubly linked list.

## 2.THEORY:

Doubly linked list is the data structure used for storing collection of nodes and the node has one data field and two fields for reference to previous node and next node in a list. The first node is the head node and its prev field is null and the last node has data and points to null. The various operations of doubly linked list are discussed below:



#### 2.1. Traverse:

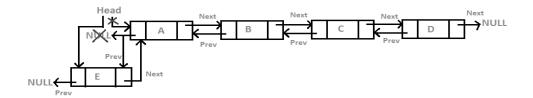
The traverse operation helps to display the content of a doubly linked list. Here, to run this operation, the temp node is kept moving to the next one and the content is displayed.

#### 2.2.Insertion:

The insertion operation inserts a node in linked list. The insertion operation in doubly linked list can be run in three different methods. They are:

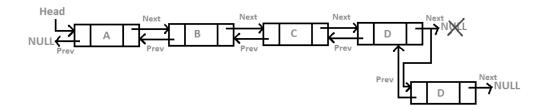
## **2.2.1.insertatbeg(int x)**:

In this method, a node is inserted at the beginning of the doubly linked list.



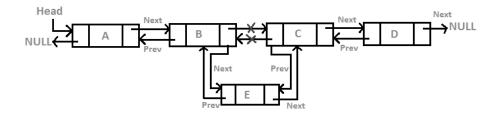
## 2.2.2.insertatlast(int x):

In this method, a node is inserted at the last of the linked list.



## 2.2.3.insertatpos(int x, int pos):

This method inserts a node at the specified position of the linked list.

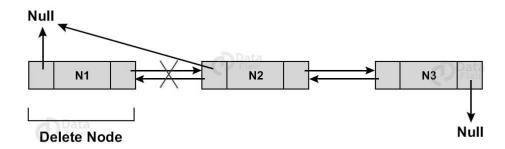


## 2.3. Deletion:

The deletion operation deletes a node from the linked list. The deletion operation in doubly linked list can also be run in three different methods. They are:

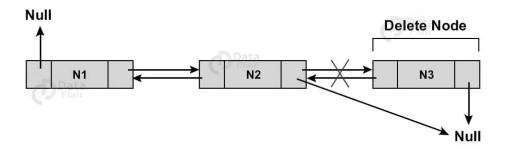
## **2.3.1.deleteatbeg()**:

In this method, a node is deleted from the very beginning of the doubly linked list.



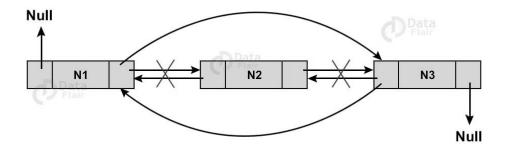
## 2.3.2.deleteatlast():

In this method, a node is deleted from the last of the doubly linked list.



## 2.3.3.deleteatpos(int pos):

In this method, a node is deleted from the specified position of the linked list.



## 2.4. Sort:

This method arranges the data of a node in either ascending or descending order.

## **3.IMPLEMENTATION:**

The following programs shows the insertion of nodes at the beginning, at last and at certain position in doubly linked list data structure in java. Similarly, the deletion of node from the beginning, from the last and from the specified position is also shown. Moreover, the source code and output for sort operation is also shown.

## 3.1.PROGRAM 1: Traverse a doubly linked list.

#### PSEUDOCODE:

## **SOURCECODE:**

```
class dlldisplay
                                                    public void display()
  class node
                                                       if(head = = null)
    int data;
                                                         System.out.println("empty list");
    node prev;
    node next;
                                                       else
    node(int data)
                                                         node temp=head;
                                                         System.out.print("null<-->");
       this.data=data;
       this.prev=null;
                                                         while(temp!=null)
       this.next=null;
                                                            System.out.print(temp.data+"<--
                                                  >");
  public node head=null;
                                                            temp=temp.next;
  public void add(int x)
                                                         System.out.print(temp);
    node\ newnode=new\ node(x);
    if(head = = null)
                                                  public static void main(String args[])
       head=newnode;
                                                       dlldisplay d=new dlldisplay();
    else
                                                       d.add(5);
                                                       d.add(10);
       head.prev=newnode;
                                                       d.add(15);
       newnode.next=head;
                                                       d.display();
       head=newnode;
```

```
A doubly linked list traversed as: null<-->15<-->10<-->5<-->null
```

**3.2 PROGRAM 2:** Insert element to a linked list either at the beginning, middle or end of the linked list.

## 3.2.1. Insert at the beginning:

### PSEUDOCODE:

```
public void insertatbeg(int x)
{
    node newnode=new node(x);
    if(head==null)
    {
        head=newnode;
    }
    else
    {
        head.prev=newnode;
        newnode.next=head;
        head=newnode;
    }
}
```

### 3.2.2. Insert at the last:

### PSEUDOCODE:

```
public void insertatlast(int x)
    {
        node newnode=new node(x);
        if(head==null)
        {
            head=newnode;
        }
        else
        {
            node temp=head;
            while(temp.next!=null)
            {
                temp=temp.next;
            }
            temp.next=newnode;
            newnode.prev=temp;
        }
    }
}
```

## 3.2.3. Insert at a specified position:

#### PSEUDOCODE:

```
public void insertatpos(int x,int pos)
                                                        for(int i=1; i < pos-
                                                 1&&temp!=null;i++) {
    node\ newnode=new\ node(x);
                                                          temp=temp.next;
    if(pos<1)
                                                        if(temp = = null)
       System.out.println("pos<1 is
                                                          System.out.println("Invalid
invalid");
                                                 position");
    else\ if(pos==1)
                                                        else if(temp.next = = null)
       if(head = = null)
                                                          temp.next=newnode;
         head=newnode;
                                                          newnode.prev=temp;
                                                        }
       else
                                                        else
         newnode.next=head;
                                                          newnode.next=temp.next;
         head.prev=newnode;
                                                          temp.next.prev=newnode;
         head=newnode;
                                                          temp.next=newnode;
                                                          newnode.prev=temp;
     else{
       node temp=head;
```

#### **SOURCE CODE OF INSERTION OPERATION:**

```
import java.util.Scanner;
class dllinsert
                                                      node\ newnode=new\ node(x);
                                                      if(head = = null)
  class node
                                                        head=newnode;
    int data;
                                                      else
    node prev;
    node next;
    node(int data)
                                                        node temp=head;
                                                        while(temp.next!=null)
       this.data=data;
       this.prev=null;
                                                           temp=temp.next;
       this.next=null;
                                                        temp.next=newnode;
                                                        newnode.prev=temp;
  public node head=null;
  public void add(int x)
                                                   public void insertatpos(int x,int pos)
    node\ newnode=new\ node(x);
    if(head = = null)
                                                      node\ newnode=new\ node(x);
                                                      if(pos<1)
       head=newnode;
                                                        System.out.println("pos<1 is
    else
                                                 invalid");
       head.prev=newnode;
                                                      else\ if(pos==1)
       newnode.next=head;
       head=newnode;
                                                        if(head = = null)
                                                           head=newnode;
  public void insertatbeg(int x)
                                                        else
    node\ newnode=new\ node(x);
                                                          newnode.next=head;
    if(head = = null)
                                                          head.prev=newnode;
                                                          head=newnode;
       head=newnode;
    else
                                                      else
       head.prev=newnode;
                                                        node temp=head;
       newnode.next=head;
                                                        for(int i=1; i < pos-
       head=newnode;
                                                 1&&temp!=null;i++)
                                                           temp=temp.next;
  public void insertatlast(int x)
                                                        if(temp = = null)
```

{	System.out.println("This is doubly
System.out.println("Invalid	linked list:");
position");	d.display();
}	int ch;
else if(temp.next==null)	do
{	{
temp.next=newnode;	System.out.println();
newnode.prev=temp;	System.out.println("Where do you
newnoue.prev=temp,	want to insert your newnode?");
else	System.out.println("At the
eise (	
	beginning? Press I");
newnode.next=temp.next;	System.out.println("At the last?
temp.next.prev=newnode;	Press 2");
temp.next=newnode;	System.out.println("At your
newnode.prev=temp;	specified position? Press 3");
}	ch=input.nextInt();
	switch(ch)
}	{
}	case 1:
	{
<pre>public void display()</pre>	System.out.println("Inserting
{	a node at beginning.");
if(head==null)	System.out.println("What is
<i>(</i>	the data of your newnode?");
System.out.println("empty list");	<pre>int a=input.nextInt();</pre>
}	d.insertatbeg(a);
else	d.display();
{	break;
node temp=head;	}
System.out.print("null<>");	case 2:
while(temp!=null)	[
{	System.out.println("Inserting
System.out.print(temp.data+"<	a node at last.");
>");	System.out.println("What is
	the data of your newnode?");
temp=temp.next;	• •
Sundania and mindle (dania)	int a=input.nextInt();
System.out.println(temp);	d.insertatlast(a);
,	d.display();
, }	break;
}	}
	case 3:
<pre>public static void main(String args[])</pre>	{
{	System.out.println("Inserting
dllinsert d=new dllinsert();	a node at specified position");
Scanner input=new	System.out.println("What is
Scanner(System.in);	the data of your newnode?");
d.add(1);	<pre>int a=input.nextInt();</pre>
d.add(2);	System.out.println("What is
d.add(3);	the position of your newnode?");
	<pre>int b=input.nextInt();</pre>

```
This is doubly linked list:
null<-->3<-->2<-->1<-->null
Where do you want to insert your newnode?
At the beginning? Press 1
At the last? Press 2
At your specified position? Press 3
Inserting a node at beginning.
What is the data of your newnode?
null<-->4<-->3<-->2<-->1<-->null
*************
Where do you want to insert your newnode?
At the beginning? Press 1
At the last? Press 2
At your specified position? Press 3
Inserting a node at last.
What is the data of your newnode?
null<-->4<-->3<-->2<-->1<-->6<-->null
***************
Where do you want to insert your newnode?
At the beginning? Press 1
At the last? Press 2
At your specified position? Press 3
Inserting a node at specified position
What is the data of your newnode?
What is the position of your newnode?
```

```
-1
pos<1 is invalid
null<-->4<-->3<-->2<-->1<-->6<-->null
Where do you want to insert your newnode?
At the beginning? Press 1
At the last? Press 2
At your specified position? Press 3
Inserting a node at specified position
What is the data of your newnode?
What is the position of your newnode?
null<-->7<-->4<-->3<-->2<-->1<-->6<-->null
Where do you want to insert your newnode?
At the beginning? Press 1
At the last? Press 2
At your specified position? Press 3
Inserting a node at specified position
What is the data of your newnode?
What is the position of your newnode?
null<-->7<-->4<-->3<-->2<-->5<-->1<-->6<-->null
Where do you want to insert your newnode?
At the beginning? Press 1
At the last? Press 2
At your specified position? Press 3
Inserting a node at specified position
What is the data of your newnode?
What is the position of your newnode?
null<-->7<-->4<-->3<-->2<-->5<-->1<-->6<-->8<-->null
Where do you want to insert your newnode?
At the beginning? Press 1
At the last? Press 2
At your specified position? Press 3
```

**3.3. PROGRAM 3:** Delete element from a linked list either from the beginning, middle or end of the linked list.

# **3.3.1. Deletion from the beginning:** PSEUDOCODE:

```
public void deleteatbeg()

{
    if(head==null)
    {
       System.out.println("Empty list");
    }
    else if(head.next==null)
    {
       head=null;
    }
    else
    {
       head=head.next;
       head.prev=null;
    }
}
```

# **3.3.2. Deletion from the last:** PSEUDOCODE:

```
public void deleteatlast()
    {
        if(head==null)
        {
            System.out.println("Empty list");
        }
        else if(head.next==null)
        {
            head=null;
        }
        else
        {
            node temp=head;
            while(temp.next.next!=null)
            {
                temp=temp.next;
            }
            temp.next=null;
        }
}
```

## **3.3.3.** Deletion from the specified position:

#### PSEUDOCODE:

```
public void deleteatpos(int pos)
                                                       else
     if(pos<1)
                                                          node temp=head;
                                                         for(int i=1; i < pos-
       System.out.println("pos<1 is
invalid");
                                                  1&&temp.next!=null;i++)
     else\ if(pos==1)
                                                            temp=temp.next;
       if(head = = null)
                                                          if(temp.next!=null)
          System.out.println("Empty list");
                                                            temp.next=temp.next.next;
                                                            temp.next.prev=temp;
       else if(head.next==null)
                                                          else{
          head=null;
                                                            System.out.println("invalid
                                                  position");
       else
          head=head.next;
          head.prev=null;
```

#### **SOURCE CODE OF DELETE OPERATION:**

```
import java.util.Scanner;
class dlldelete
                                                    public void deleteatlast()
  class node
                                                       if(head = = null)
    int data;
                                                          System.out.println("Empty list");
    node prev;
    node next;
                                                       else if(head.next==null)
    node(int data)
                                                          head=null;
       this.data=data;
       this.prev=null;
                                                       else
       this.next=null;
                                                          node temp=head;
                                                          while(temp.next.next!=null)
  public node head=null;
                                                            temp=temp.next;
  public void add(int x)
                                                          temp.next=null;
    node\ newnode=new\ node(x);
    if(head = = null)
                                                    public void deleteatpos(int pos)
       head=newnode;
    else
                                                       if(pos<1)
       head.prev=newnode;
                                                          System.out.println("pos<1 is
       newnode.next=head;
                                                  invalid");
       head=newnode;
                                                       else\ if(pos==1)
                                                          if(head = = null)
  public void deleteatbeg()
                                                            System.out.println("Empty list");
    if(head = = null)
                                                          else if(head.next==null)
       System.out.println("Empty list");
                                                            head=null;
    else if(head.next==null)
                                                          else
       head=null;
                                                            head=head.next;
    else
                                                            head.prev=null;
       head=head.next;
       head.prev=null;
                                                       else
                                                          node temp=head;
```

for(int i=1; i < pos-	System.out.println("This is doubly		
1&&temp.next!=null;i++)	linked list:");		
<u></u>	d.display();		
temp=temp.next;	int ch;		
}	do		
if(temp.next!=null)	{		
<i>f</i>	System.out.println();		
temp.next=temp.next.next;	System.out.println("From where do		
temp.next.prev=temp;	you want to delete your newnode?");		
}	System.out.println("From the		
else	beginning? Press 1");		
<i>[</i>	System.out.println("From the last?		
System.out.println("invalid	Press 2");		
	**		
position");	System.out.println("From the specified position? Press 3");		
<i>J</i>			
<i>}</i>	ch=input.nextInt();		
<i>}</i>	switch(ch)		
11 1 1. 1	{		
public void display()	case 1:		
{			
if(head==null)	System.out.println("Deleteing		
{	a node from the beginning.");		
System.out.println("empty list");	d. delete at beg();		
}	d.display();		
else	System.out.println("******		
{	**************************************		
node temp=head;	break;		
System.out.print("null<>");	}		
while(temp!=null)	case 2:		
{	{		
System.out.print(temp.data+"<	System.out.println("Delete a		
>");	node at last.");		
temp=temp.next;	<pre>d.deleteatlast();</pre>		
}	d.display();		
System.out.println(temp);	System.out.println("******* *****************************		
}	break;		
,	}		
,	case 3:		
<pre>public static void main(String args[])</pre>	{		
{	System.out.println("Delete a		
dlldelete d=new dlldelete();	node at specified position");		
Scanner input=new	System.out.println("What is		
Scanner(System.in);	the position of your newnode?");		
d.add(1);	int b=input.nextInt();		
a.aaa(1); d.add(2);	d.deleteatpos(b);		
	* * * * * * * * * * * * * * * * * * * *		
d.add(3);	d.display();		
d.add(4);	break;		
d.add(5);	f LC L		
d.add(6);	default:		

```
This is doubly linked list:
null<-->6<-->5<-->4<-->3<-->2<-->1<-->null
From where do you want to delete your newnode?
From the beginning? Press 1
From the last? Press 2
From the specified position? Press 3
Deleteing a node from the beginning.
null<-->5<-->4<-->3<-->2<-->1<-->null
********************
From where do you want to delete your newnode?
From the beginning? Press 1
From the last? Press 2
From the specified position? Press 3
Delete a node at last.
null<-->5<-->4<-->3<-->2<-->null
**********************
From where do you want to delete your newnode?
From the beginning? Press 1
From the last? Press 2
From the specified position? Press 3
Delete a node at specified position
What is the position of your newnode?
pos<1 is invalid</pre>
null<-->5<-->4<-->3<-->2<-->null
From where do you want to delete your newnode?
From the beginning? Press 1
From the last? Press 2
From the specified position? Press 3
```

```
Delete a node at specified position
What is the position of your newnode?
null<-->5<-->3<-->2<-->null
From where do you want to delete your newnode?
From the beginning? Press 1
From the last? Press 2
From the specified position? Press 3
Delete a node at specified position
What is the position of your newnode?
invalid position
null<-->5<-->3<-->2<-->null
From where do you want to delete your newnode?
From the beginning? Press 1
From the last? Press 2
From the specified position? Press 3
Enter your choice from 1 to 3.
```

#### **3.4. PROGRAM 4:** Sort the element of the linked list.

#### PSEUDOCODE:

```
public void sort() {
    node temp;
                                                             int t=temp.data;
    node last=null;
                                                             temp.data=temp.next.data;
     int swap;
                                                             temp.next.data=t;
     do{
                                                             swap=1;
       swap=0;
       temp=head;
                                                          temp=temp.next;
       while(temp.next!=last)
                                                        last=temp;
         if(temp.data>temp.next.data)
                                                     }while(swap!=0);
```

## **SOURCECODE:**

```
class dllsort
                                                      node temp;
                                                      node last=null;
  class node
                                                      int swap;
                                                      do
    int data;
    node next:
                                                        swap=0;
    node prev;
                                                        temp=head;
    node(int data)
                                                        while(temp.next!=last)
                                                           if(temp.data>temp.next.data)
       this.data=data;
       this.next=null;
       this.prev=null;
                                                             int t=temp.data;
                                                             temp.data=temp.next.data;
                                                             temp.next.data=t;
  public node head=null;
                                                             swap=1;
  public void add(int x)
                                                           temp=temp.next;
    node\ newnode=new\ node(x);
                                                        last=temp;
    if(head = = null)
                                                      }while(swap!=0);
       head=newnode;
                                                   public void display()
    else
                                                      node temp=head;
       newnode.next=head;
                                                      System.out.print("null<-->");
                                                      while(temp!=null)
       head.prev=newnode;
       head=newnode;
                                                        System.out.print(temp.data+"<--
                                                 >");
                                                        temp=temp.next;
  public void sort()
                                                      System.out.println(temp);
```

```
The linked list before sorting:
null<-->3<-->2<-->1<-->null

The linked list after sorting:
null<-->1<-->2<-->3<-->null
```

## 4. OUTPUT AND DISCUSSION:

- ✓ In this experiment, the output of program 1 shows the traverse of a node to display the data of the node in the doubly linked list.
- ✓ In the output of program 2, it can be seen that the insertion of a node is done at the beginning of the linked list, after then, when the user press 2, a node is inserted at the last and then when the user press option 3, a node is inserted at a specified position.
- ✓ The output of the Program 3 shows the deletion of a node from the beginning, from the last and from the specified position.
- ✓ In program 4, the sorting program for linked list is done in which the data of the doubly linked list are arranged in ascending order.

## **5. CONCLUSION:**

A Java program to traverse, insert a node at first, last, certain position, delete a node at beginning, at last and at certain position and sorting of the data of doubly linked list is successfully run.

## **EXPERIMENT: 3**

## TITLE:

A java program to run various operations of circular singly linked list.

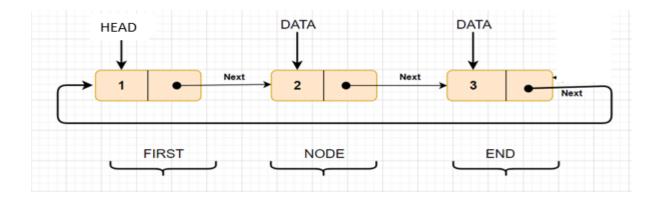
## 1.OBJECTIVE:

- ✓ To insert a node at beginning, at last and at certain position and display it.
- ✓ To delete a node from the beginning, from last and from certain position.
- ✓ To count the nodes in the circular singly linked list.

## 2.THEORY:

Circular singly linked list is the data structure used for storing collection of nodes and the node has one data field and a field for reference to next node in a list. The first node is the head node and it has data and points to next node and the last node has data and points to head.

The various operations of circular singly linked list are discussed below:



#### 2.1. Traverse:

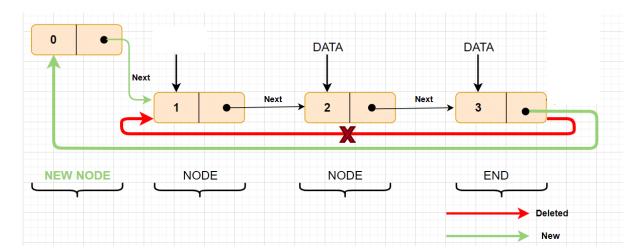
The traverse operation helps to display the content of a circular singly linked list. Here, to run this operation, the temp node is kept moving to the next one and the content is displayed.

#### 2.2.Insertion:

The insertion operation inserts a node in linked list. The insertion operation in circular singly linked list can be run in three different methods. They are:

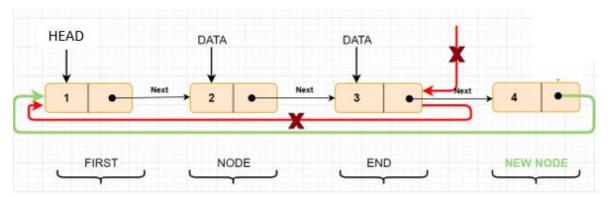
## **2.2.1.insertatbeg(int x)**:

In this method, a node is inserted at the beginning of the circular singly linked list.



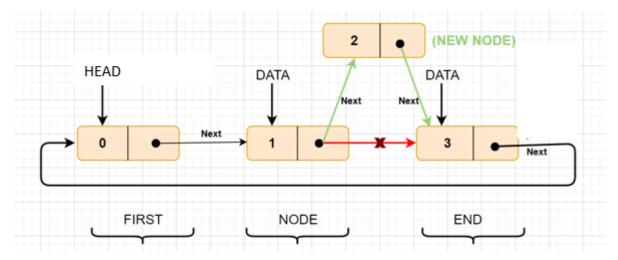
## 2.2.2.insertatlast(int x):

In this method, a node is inserted at the last of the linked list.



## 2.2.3.insertatpos(int x, int pos):

This method inserts a node at the specified position of the linked list.

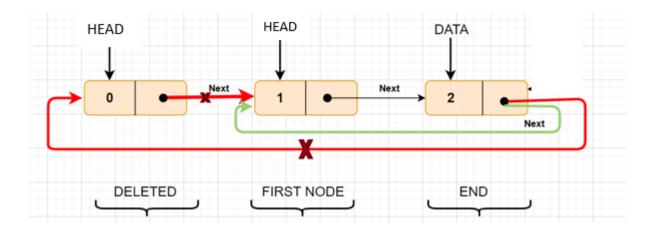


## 2.3. Deletion:

The deletion operation deletes a node from the linked list. The deletion operation in circular singly linked list can also be run in three different methods. They are:

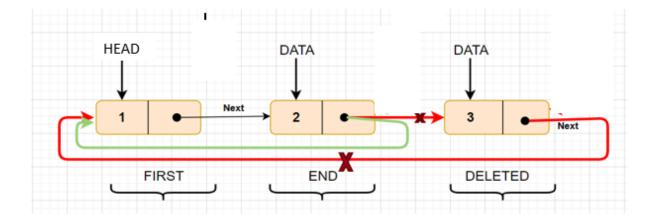
## **2.3.1.deleteatbeg()**:

In this method, a node is deleted from the very beginning of the circular singly linked list.



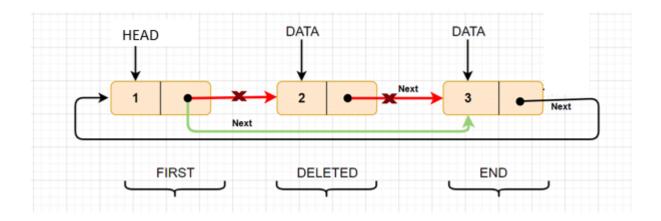
# 2.3.2.deleteatlast():

In this method, a node is deleted from the last of the circular singly linked list.



# 2.3.3.deleteatpos(int pos):

In this method, a node is deleted from the specified position of the linked list.



## **2.4. Count:**

This method counts the number of nodes in the given circular singly linked list.

# **3.IMPLEMENTATION:**

The following programs shows the insertion of nodes at the beginning, at last and at certain position in circular singly linked list data structure in java. Similarly, the deletion of node from the beginning, from the last and from the specified position is also shown. Moreover, the source code and output for count operation is also shown.

# **3.1.PROGRAM 1**: Traverse a circular singly linked list.

### **PSEUDOCODE:**

## **SOURCECODE:**

```
class cslltraverse
                                                 public void display()
  class node
                                                      if(head = = null)
    int data:
                                                         System.out.println("empty list");
    node next;
    node(int data)
                                                      else {
                                                         node temp=head;
       this.data=data;
                                                         while(temp.next!=head)
       this.next=null;
                                                           System.out.print(temp.data+"--
                                                  >");
  public node head=null;
                                                           temp=temp.next;
  public void add(int x)
                                                         System.out.print(temp.data);
    node\ newnode=new\ node(x);
    if(head = = null){
                                                      node temp=head;
       head=newnode:
       head.next=head;
                                                    public static void main(String args[])
    }
    else {
       node temp=head;
                                                      cslltraverse d=new cslltraverse();
       while(temp.next!=head)
                                                      System.out.println("A circular singly
                                                 linked list traversed as:");
                                                      d.add(5);
         temp=temp.next;
                                                      d.add(10);
       temp.next=newnode;
                                                      d.display();
       newnode.next=head;
       head=newnode;
```

```
A circular singly linked list traversed as:
```

**3.2 PROGRAM 2:** Insert element to a linked list either at the beginning, middle or end of the linked list.

# **3.2.1. Insert at the beginning:** PSEUDOCODE:

```
public void insertatbeg(int x)
{
    node newnode=new node(x);
    if(head==null)
    {
        head=newnode;
        head.next=head;
    }
    else
    {
        node temp=head;
        while(temp.next!=head)
        {
            temp=temp.next;
        }
        temp.next=newnode;
        newnode.next=head;
        head=newnode;
    }
}
```

# **3.2.2. Insert at the last:** PSEUDOCODE:

```
public void insertatlast(int x)
{
    node newnode=new node(x);
    if(head==null)
    {
        head=newnode;
        head.next=head;
    }
    else
    {
        node temp=head;
        while(temp.next!=head)
        {
            temp=temp.next;
        }
        temp.next=newnode;
        newnode.next=head;
    }
}
```

## 3.2.3. Insert at a specified position:

## **PSEUDOCODE**:

```
public void insertatpos(int x,int pos) {
                                                     else if(pos>(count()+1)) {
    node\ newnode=new\ node(x);
                                                          System.out.println("invalid
    if(pos<1){
                                                position");
       System.out.println("pos<1 is
invalid");
                                                     else{
                                                       node temp=head;
    else\ if(pos==1) {
                                                       int i;
       if(head==null){
                                                       for(i=1;i<pos-
         head=newnode;
                                                1\&\&i <= count(); i++)
         head.next=head;
                                                          temp=temp.next;
       else{
                                                       if(temp.next!=head){
         node temp=head;
                                                          newnode.next=temp.next;
         while(temp.next!=head){
                                                          temp.next=newnode;
           temp=temp.next;
                                                       else {
         temp.next=newnode;
                                                          temp.next=newnode;
         newnode.next=head;
                                                          newnode.next=head;
         head=newnode;
```

## **SOURCE CODE OF INSERTION OPERATON:**

```
import java.util.Scanner;
                                                          temp=temp.next;
class csllinsert
                                                        temp.next=newnode;
  class node
                                                        newnode.next=head;
                                                        head=newnode;
    int data:
    node next;
    node(int data)
                                                   public void insertatlast(int x)
       this.data=data;
                                                     node\ newnode=new\ node(x);
       this.next=null;
                                                     if(head = = null)
  public node head=null;
                                                        head=newnode:
                                                        head.next=head;
  public void add(int x)
                                                     else
    node\ newnode=new\ node(x);
    if(head = = null)
                                                        node temp=head;
                                                        while(temp.next!=head)
       head=newnode;
       head.next=head;
                                                          temp=temp.next;
    else
                                                        temp.next=newnode;
                                                       newnode.next=head:
       node temp=head;
       while(temp.next!=head)
         temp=temp.next;
       temp.next=newnode;
                                                  public void insertatpos(int x,int pos)
       newnode.next=head;
       head=newnode;
                                                     node\ newnode=new\ node(x);
                                                     if(pos<1)
                                                        System.out.println("pos<1 is
  public void insertatbeg(int x)
                                                invalid");
    node\ newnode=new\ node(x);
                                                     else\ if(pos==1)
    if(head = = null)
                                                        if(head = = null)
       head=newnode;
       head.next=head;
                                                          head=newnode;
                                                          head.next=head;
    else
                                                        else
       node temp=head;
       while(temp.next!=head)
                                                          node temp=head;
                                                          while(temp.next!=head)
```

```
{
                                                          System.out.print(temp.data);
            temp=temp.next;
          temp.next=newnode;
          newnode.next=head;
          head=newnode;
                                                    public int count()
     else if(pos>(count()+1))
                                                       int c=0;
                                                       if(head = = null)
          System.out.println("invalid
position");
                                                          return c;
     else
                                                       else
       node temp=head;
                                                          node temp=head;
       int i;
                                                          while(temp.next!=head)
       for(i=1;i< pos-
1\&\&i < = count(); i++)
                                                            c=c+1;
                                                            temp=temp.next;
          temp=temp.next;
                                                          return c+1;
       if(temp.next!=head)
         newnode.next=temp.next;
          temp.next=newnode;
                                                    public static void main(String args[])
                                                       csllinsert s=new csllinsert();
       else
                                                       System.out.println("A circular singly
                                                  linked list is given as:");
          temp.next=newnode;
          newnode.next=head;
                                                       s.add(5);
                                                       s.add(10);
                                                       s.add(15);
                                                       s.display();
                                                       int ch;
  public void display()
                                                       do{
                                                       System.out.println();
     if(head = = null)
                                                       System.out.println("press 1 for
                                                  insertion at beginning");
       System.out.println("empty list");
                                                       System.out.println("press 2 for
                                                  insertion at last");
     else
                                                       System.out.println("press 3 for
                                                  insertion at position");
       node temp=head;
                                                       System.out.println("enter your
       while(temp.next!=head)
                                                  choice");
                                                       Scanner input=new
          System.out.print(temp.data+"--
                                                  Scanner(System.in);
>");
                                                       ch=input.nextInt();
                                                       switch(ch)
          temp=temp.next;
```

```
s.insertatpos(1,1);
     {
       case 1:
                                                               s.display();
                                                               System.out.println();
          s.insertatbeg(9);
                                                               System.out.println("Insertion at
                                                    position 2");
          s.display();
          break;
                                                               s.insertatpos(3,2);
                                                               s.display();
                                                               System.out.println();
                                                               System.out.println("Insertion at
       case 2:
                                                    position 10");
                                                               s.insertatpos(10,10);
          s.insertatlast(11);
          s.display();
                                                               s.display();
          break;
                                                               break;
       case 3:
                                                            default:
          System.out.println("Insertion at
                                                               System.out.println("please enter
pos<1");
                                                    your choice properly");
          s.insertatpos(1, -1);
          s.display();
          System.out.println();
                                                         } while(ch <= 3);
          System.out.println("Insertion at
pos=1");
```

```
A circular singly linked list is given as:

15-->10-->5

press 1 for insertion at beginning
press 2 for insertion at last
press 3 for insertion at position
enter your choice

1
9-->15-->10-->5

press 1 for insertion at beginning
press 2 for insertion at last
press 3 for insertion at position
enter your choice
2
9-->15-->10-->5-->11
```

```
press 1 for insertion at beginning
press 2 for insertion at last
press 3 for insertion at position
enter your choice
Insertion at pos<1
pos<1 is invalid
9-->15-->10-->5-->11
Insertion at pos=1
1-->9-->15-->10-->5-->11
Insertion at position 2
1-->3-->9-->15-->10-->5-->11
Insertion at position 10
invalid position
1-->3-->9-->15-->10-->5-->11
press 1 for insertion at beginning
press 2 for insertion at last
press 3 for insertion at position
enter your choice
please enter your choice properly
```

**3.3. PROGRAM 3:** Delete element from a linked list either from the beginning, middle or end of the linked list.

# **3.3.1. Deletion from the beginning:** PSEUDOCODE:

```
public void deleteatbeg()
{
    if(head==null){
        System.out.println("Empty list");
    }
    else if(head.next==null)
    {
        head=null;
    }
    else
    {
        node temp=head;
        while(temp.next!=head)
        {
            temp=temp.next;
        }
        temp.next=head.next;
        head=head.next;
    }
}
```

# **3.3.2. Deletion from the last:** PSEUDOCODE:

```
public void deleteatlast()
    {
        if(head==null)
        {
            System.out.println("Empty list");
        }
        else if(head.next==null)
        {
            head=null;
        }
        else
        {
            node temp=head;
            while(temp.next.next!=head)
            {
                temp=temp.next;
            }
            temp.next=head;
        }
}
```

## 3.3.3. Deletion from the specified position:

#### PSEUDOCODE:

```
temp.next=head.next;
public void deleteatpos(int pos)
                                                           head=head.next;
  {
     if(pos<1){
       System.out.println("pos<1 is
invalid");
                                                      else{
                                                         node temp=head;
     else\ if(pos==1){
                                                        for(int i=1; i < pos-
       if(head==null){
                                                  1\&\&temp.next!=head;i++)
          System.out.println("Empty
list ");
                                                           temp=temp.next;
       else if(head.next==null)
                                                         if(temp.next!=head)
          head=null;
                                                           temp.next=temp.next.next;
                                                         else{
       else{
          node temp=head;
                                                           System.out.println("invalid
          while(temp.next!=head)
                                                 position");
            temp=temp.next;
```

### **SOURCE CODE FOR DELETION OPERATION:**

```
import java.util.Scanner;
class cslldelete
                                                        temp.next=head.next;
                                                        head=head.next;
  class node
    int data:
    node next;
                                                   public void deleteatlast()
    node(int data)
                                                      if(head==null){
                                                        System.out.println("Empty list");
       this.data=data;
       this.next=null;
                                                      else if(head.next==null)
  public node head=null;
                                                        head=null;
  public void add(int x)
                                                      else {
                                                        node temp=head;
    node\ newnode=new\ node(x);
                                                        while(temp.next.next!=head)
    if(head==null){
       head=newnode;
                                                           temp=temp.next;
       head.next=head;
    }
                                                        temp.next=head;
    else {
       node temp=head;
       while(temp.next!=head)
                                                    public void deleteatpos(int pos)
         temp=temp.next;
                                                      if(pos<1) {
       temp.next=newnode;
                                                        System.out.println("pos<1 is
       newnode.next=head;
                                                 invalid");
       head=newnode;
                                                      else\ if(pos==1)
                                                        if(head = = null)
  public void deleteatbeg()
                                                           System.out.println("Empty list");
    if(head = = null)
                                                        else if(head.next==null)
       System.out.println("Empty list");
                                                           head=null;
    else if(head.next==null)
                                                        else {
       head=null;
                                                           node temp=head;
                                                           while(temp.next!=head) {
                                                             temp=temp.next;
    else {
       node temp=head;
       while(temp.next!=head)
                                                           temp.next=head.next;
                                                           head=head.next;
         temp=temp.next;
```

```
int ch;
     }
     else {
                                                       do{
       node temp=head;
                                                          System.out.println();
       for(int i=1; i < pos-
                                                          System.out.println("From where
                                                  do you want to delete your newnode?");
1\&\&temp.next!=head;i++)
                                                          System.out.println("From the
                                                  beginning? Press 1");
          temp=temp.next;
                                                          System.out.println("From the last?
       if(temp.next!=head)
                                                  Press 2 ");
                                                          System.out.println("From the
                                                  specified position? Press 3");
          temp.next=temp.next.next;
                                                          ch=input.nextInt();
       else {
                                                          switch(ch)
          System.out.println("invalid
position");
                                                            case 1:
                                                               System.out.println("Deleteing
                                                  a node from the beginning.");
                                                               d.deleteatbeg();
  public void display()
                                                               d.display();
                                                               break;
     if(head==null){
       System.out.println("empty list");
                                                            case 2:
     else{
                                                               System.out.println("Delete a
       node temp=head;
                                                  node at last.");
       while(temp.next!=head) {
                                                               d.deleteatlast();
          System.out.print(temp.data+"←
                                                               d.display();
>");
                                                               break;
          temp=temp.next;
                                                            case 3:
       System.out.print(temp.data);
                                                               System.out.println("Delete a
                                                  node at specified position");
                                                               System.out.println("What is
                                                  the position of your newnode?");
  public static void main(String args[])
                                                               int b=input.nextInt();
                                                               d.deleteatpos(b);
     cslldelete d=new cslldelete();
                                                               d.display();
     Scanner input=new
                                                               break;
Scanner(System.in);
     d.add(1);
                                                            default:{
                                                               System.out.println("Enter
     d.add(2);
     d.add(3);
                                                  your choice from 1 to 3.");
     d.add(4);
     d.add(5);
     d.add(6);
                                                       \}while(ch<=3);
     System.out.println("This is circular
singly linked list:");
    d.display();
```

```
This is circular singly linked list:
6<-->5<-->4<-->3<-->2
From where do you want to delete your newnode?
From the beginning? Press 1
From the last? Press 2
From the specified position? Press 3
Deleting a node from the beginning.
5<-->4<-->3<-->2<-->1
From where do you want to delete your newnode?
From the beginning? Press 1
From the last? Press 2
From the specified position? Press 3
Deletion of a node at last.
5<-->4<-->3<-->2
From where do you want to delete your newnode?
From the beginning? Press 1
From the last? Press 2
From the specified position? Press 3
Delete a node at specified position
What is the position of your newnode?
-1
pos<1 is invalid
5<-->4<-->3<-->2
What is the position of your newnode?
invalid position
5<-->4<-->2
What is the position of your newnode?
```

```
1
4<-->3<-->2
What is the position of your newnode?
2
4<-->2
From where do you want to delete your newnode?
From the beginning? Press 1
From the last? Press 2
From the specified position? Press 3
4
Enter your choice from 1 to 3.
```

## **3.4. PROGRAM 4:** Count the element of the linked list.

### PSEUDOCODE:

### SOURCECODE:

```
class csllcount
                                                   public int count()
  class node
                                                      int c=0:
    int data;
                                                      if(head = = null)
    node next;
    node(int data)
                                                        return c;
       this.data=data;
                                                      else
       this.next=null;
                                                        node temp=head;
                                                        while(temp.next!=head)
  public node head=null;
                                                           c=c+1;
  public void add(int x)
                                                           temp=temp.next;
    node\ newnode=new\ node(x);
                                                        return c+1;
    if(head = = null)
       head=newnode;
       head.next=head;
                                                   public void display()
    else
                                                      if(head = = null)
       node temp=head;
                                                        System.out.println("empty list");
       while(temp.next!=head)
                                                      else
         temp=temp.next;
                                                        node temp=head;
                                                        while(temp.next!=head)
       temp.next=newnode;
       newnode.next=head;
       head=newnode;
                                                           System.out.print(temp.data+"--
                                                 >");
```

```
A circular singly linked list is given as:
15-->10-->5
The number of nodes in the linked list are:3
```

## 4. OUTPUT AND DISCUSSION:

- ✓ In this experiment, the output of program 1 shows the traverse of a node to display the data of the node in the circular singly linked list.
- ✓ In the output of program 2, it can be seen that the insertion of a node is done at the beginning of the linked list, after then, when the user press 2, a node is inserted at the last and then when the user press option 3, a node is inserted at a specified position.
- ✓ The output of the Program 3 shows the deletion of a node from the beginning, from the last and from the specified position.
- ✓ In program 4, the number of nodes are counted in the given circular singly linked list...

## 5. CONCLUSION:

A Java program to traverse, insert a node at first, last, certain position, delete a node at beginning, at last and at certain position and counting of the nodes in circular singly linked list is successfully run.

# **EXPERIMENT: 4**

## TITLE:

A java program to run various operations of circular doubly linked list.

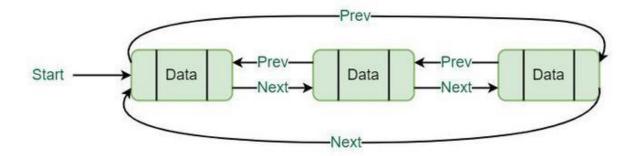
## 1.OBJECTIVE:

- ✓ To insert a node at beginning, at last and at certain position and display it.
- ✓ To delete a node from the beginning, from last and from certain position.
- ✓ To count the number of nodes in the linked list.

## 2.THEORY:

Circular doubly linked list is the data structure used for storing collection of nodes and the node has one data field and two fields for reference to previous node and next node in a list. The first node is the head node and its prev field points to the last node and the last node has data and points to head.

The various operations of circular doubly linked list are discussed below:



# 2.1. Traverse:

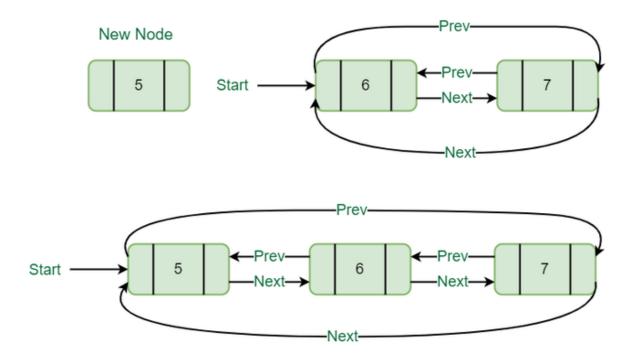
The traverse operation helps to display the content of a circular doubly linked list. Here, to run this operation, the temp node is kept moving to the next one and the content is displayed.

## 2.2.Insertion:

The insertion operation inserts a node in linked list. The insertion operation in circular doubly linked list can be run in three different methods. They are:

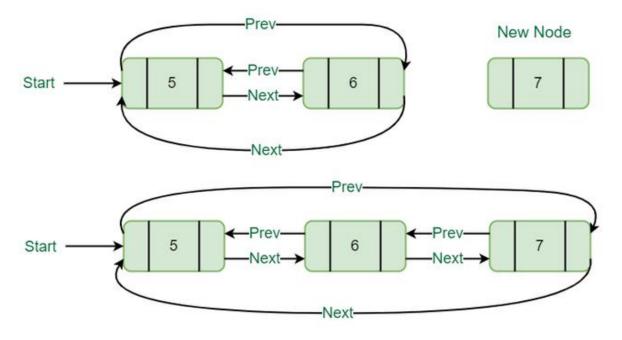
# **2.2.1.insertatbeg(int x)**:

In this method, a node is inserted at the beginning of the circular doubly linked list.



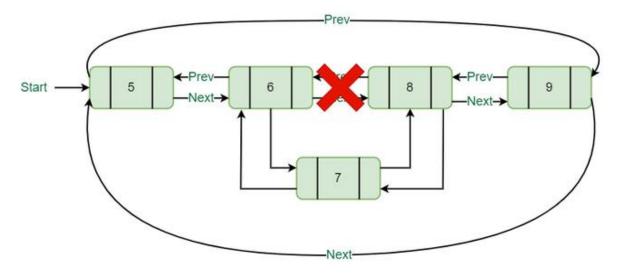
# **2.2.2.insertatlast(int x):**

In this method, a node is inserted at the last of the linked list.



# 2.2.3.insertatpos(int x, int pos):

This method inserts a node at the specified position of the linked list.

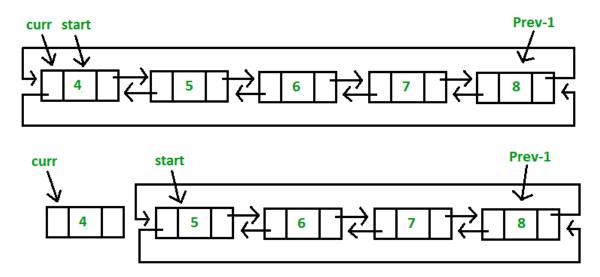


# 2.3. Deletion:

The deletion operation deletes a node from the linked list. The deletion operation in circular singly linked list can also be run in three different methods. They are:

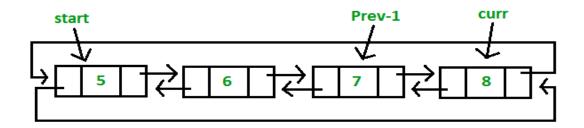
# 2.3.1.deleteatbeg():

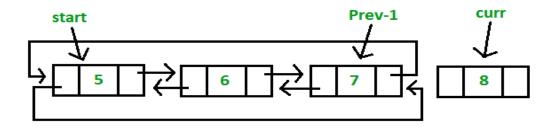
In this method, a node is deleted from the very beginning of the circular singly linked list.



# 2.3.2.deleteatlast():

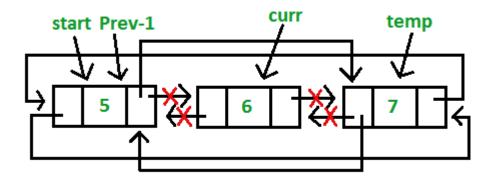
In this method, a node is deleted from the last of the circular singly linked list.





## 2.3.3.deleteatpos(int pos):

In this method, a node is deleted from the specified position of the linked list.



## **2.4. Count:**

This method counts the number of nodes in the circular doubly linked list.

## **3.IMPLEMENTATION:**

The following programs shows the insertion of nodes at the beginning, at last and at certain position in circular doubly linked list data structure in java. Similarly, the deletion of node from the beginning, from the last and from the specified position is also shown. Moreover, the source code and output for count operation is also shown.

# **3.1.PROGRAM 1**: Traverse a circular doubly linked list.

#### PSEUDOCODE:

## **SOURCECODE:**

```
class cdlltraverse{
  class node{
    int data;
                                                   public void display(){
    node next;
                                                      if(head==null){
                                                        System.out.println("empty list");
    node prev;
    node(int data){
       this.data=data;
                                                      else {
       this.next=null:
                                                        node temp=head;
       this.prev=null;
                                                        while(temp.next!=head)
                                                           System.out.print(temp.data+"<--
                                                 >");
  public node head=null;
                                                           temp=temp.next;
  public void add(int x){
    node\ newnode=new\ node(x);
                                                        System.out.print(temp.data);
    if(head = = null){
       head=newnode;
       head.prev=head;
       head.next=head;
                                                   public static void main(String args[]){
                                                      cdlltraverse d=new cdlltraverse();
    }
    else {
                                                      System.out.println("A circular doubly
       node temp=head.prev;
                                                 linked list traversed as:");
       temp.next=newnode;
                                                      d.add(5);
       newnode.next=head;
                                                      d.add(10);
       newnode.prev=temp;
                                                      d.add(15);
       head.prev=newnode;
                                                      d.display();
       head=newnode:
```

```
A circular doubly linked list traversed as: 15<-->10<-->5
```

**3.2 PROGRAM 2:** Insert element to a linked list either at the beginning, middle or end of the linked list.

# **3.2.1. Insert at the beginning:** PSEUDOCODE:

```
public void insertatbeg(int x)

{
    node newnode=new node(x);
    if(head==null)
    {
        head=newnode;
        head.prev=head;
        head.next=head;
    }
    else {
        node temp=head.prev;
        temp.next=newnode;
        newnode.next=head;
        newnode.prev=temp;
        head.prev=newnode;
        head=newnode;
    }
}
```

# **3.2.2. Insert at the last:** PSEUDOCODE:

```
public void insertatlast(int x)
{
    node newnode=new node(x);
    if(head==null)
    {
        head=newnode;
        head.prev=head;
        head.next=head;
    }
    else
    {
        node temp=head.prev;
        temp.next=newnode;
        newnode.prev=temp;
        newnode.next=head;
        head.prev=newnode;
}
```

## 3.2.3. Insert at a specified position:

## PSEUDOCODE:

```
public void insertatpos(int x,int pos) {
                                                       System.out.println("invalid
node\ newnode=new\ node(x);
                                                position");
    if(pos<1) {
       System.out.println("pos<1 is
                                                    else {
invalid");
                                                       node temp=head;
                                                       int i;
    else if(pos==1) {
                                                      for(i=1;i<pos-
       if(head==null) {
                                                1\&\&i <= count(); i++) 
         head=newnode;
                                                         temp=temp.next;
         head.prev=head;
         head.next=head;
                                                       if(temp.next!=head) {
                                                         newnode.next=temp.next;
       else{
                                                         temp.next.prev=newnode;
         node temp=head.prev;
                                                         temp.next=newnode;
         temp.next=newnode;
                                                         newnode.prev=temp;
         newnode.next=head;
         newnode.prev=temp;
                                                       else{
         head.prev=newnode;
                                                         temp.next=newnode;
         head=newnode:
                                                         newnode.prev=temp;
                                                         newnode.next=head;
                                                         head.prev=newnode;
    else if(pos>(count()+1)) {
                                                      } }
                                                  }
```

## **SOURCECODE OF INSERTION OPERATION:**

```
import java.util.Scanner;
                                                       node temp=head.prev;
class cdllinsert
                                                       temp.next=newnode;
                                                       newnode.next=head;
  class node
                                                       newnode.prev=temp;
                                                       head.prev=newnode;
    int data;
                                                       head=newnode;
    node next;
    node prev;
    node(int data)
                                                  public void insertatlast(int x)
       this.data=data;
       this.next=null;
                                                     node\ newnode=new\ node(x);
       this.prev=null;
                                                     if(head = = null)
                                                       head=newnode;
  public node head=null;
                                                       head.prev=head;
                                                       head.next=head;
  public void add(int x)
                                                     else
    node\ newnode=new\ node(x);
    if(head = = null)
                                                       node temp=head.prev;
                                                       temp.next=newnode;
       head=newnode;
                                                       newnode.prev=temp;
       head.prev=head;
                                                       newnode.next=head;
       head.next=head;
                                                       head.prev=newnode;
    else
       node temp=head.prev;
       temp.next=newnode;
       newnode.next=head;
                                                  public void insertatpos(int x,int pos)
       newnode.prev=temp;
       head.prev=newnode;
                                                     node\ newnode=new\ node(x);
       head=newnode;
                                                     if(pos<1)
                                                       System.out.println("pos<1 is
                                                invalid");
  public void insertatbeg(int x)
                                                     else\ if(pos==1)
    node\ newnode=new\ node(x);
    if(head = = null)
                                                       if(head = = null)
       head=newnode;
                                                          head=newnode;
       head.prev=head;
                                                          head.prev=head;
       head.next=head;
                                                         head.next=head;
    else
                                                       else
```

```
node temp=head.prev;
                                                           System.out.print(temp.data+"<--
         temp.next=newnode;
                                                  >");
         newnode.next=head;
                                                           temp=temp.next;
         newnode.prev=temp;
         head.prev=newnode;
                                                         System.out.print(temp.data);
         head=newnode;
                                                      node temp=head;
     else if(pos>(count()+1))
       System.out.println("invalid
                                                    public int count()
position");
                                                      int c=0;
     else
                                                      if(head = = null)
       node temp=head;
                                                         return c;
       int i;
       for(i=1;i< pos-
                                                      else
1\&\&i < = count(); i++)
                                                         node temp=head;
                                                         while(temp.next!=head)
         temp=temp.next;
       if(temp.next!=head)
                                                           c=c+1;
                                                           temp=temp.next;
         newnode.next=temp.next;
         temp.next.prev=newnode;
                                                         return c+1;
         temp.next=newnode;
         newnode.prev=temp;
       else
                                                    public static void main(String args[])
         temp.next=newnode;
                                                      cdllinsert s=new cdllinsert();
         newnode.prev=temp;
                                                      System.out.println("A circular doubly
         newnode.next=head;
                                                 linked list is given as:");
         head.prev=newnode;
                                                      s.add(5);
                                                      s.add(10);
                                                      s.add(15);
                                                      s.display();
  public void display()
                                                      int ch;
                                                      do{
     if(head = = null)
                                                      System.out.println();
                                                      System.out.println("press 1 for
                                                 insertion at beginning");
       System.out.println("empty list");
                                                      System.out.println("press 2 for
                                                 insertion at last");
     else
                                                      System.out.println("press 3 for
       node temp=head;
                                                 insertion at position");
                                                      System.out.println("enter your
       while(temp.next!=head)
                                                 choice");
```

```
Scanner input=new
                                                              System.out.println("Insertion at
                                                   pos=1");
Scanner(System.in);
     ch=input.nextInt();
                                                              s.insertatpos(1,1);
     switch(ch)
                                                              s.display();
                                                              System.out.println();
                                                              System.out.println("Insertion at
       case 1:
                                                   position 2");
          s.insertatbeg(9);
                                                              s.insertatpos(3,2);
          s.display();
                                                              s.display();
          break;
                                                              System.out.println();
                                                              System.out.println("Insertion at
                                                   position 10");
       case 2:
                                                              s.insertatpos(10,10);
                                                              s.display();
          s.insertatlast(11);
                                                              break;
          s.display();
          break;
                                                           default:
                                                              System.out.println("please enter
       case 3:
                                                   your choice properly");
          System.out.println("Insertion at
pos<1");
          s.insertatpos(1, -1);
                                                        \}while(ch<=3);
          s.display();
          System.out.println();
```

```
A circular doubly linked list is given as:

15<-->10<-->5

press 1 for insertion at beginning
press 2 for insertion at last
press 3 for insertion at position
enter your choice

1

9<-->15<-->5

press 1 for insertion at beginning
press 2 for insertion at last
press 3 for insertion at position
enter your choice
2

9<-->15<-->10<-->5<-->11
```

```
press 1 for insertion at beginning
press 2 for insertion at last
press 3 for insertion at position
enter your choice
Insertion at pos<1</pre>
pos<1 is invalid
9<-->15<-->10<-->5<-->11
Insertion at pos=1
1<-->9<-->15<-->10<-->5<-->11
Insertion at position 2
1<-->3<-->9<-->15<-->10<-->5<-->11
Insertion at position 10
invalid position
1<-->3<-->9<-->15<-->10<-->5<-->11
press 1 for insertion at beginning
press 2 for insertion at last
press 3 for insertion at position
enter your choice
please enter your choice properly
```

**3.3. PROGRAM 3:** Delete element from a linked list either from the beginning, middle or end of the linked list.

# **3.3.1. Deletion from the beginning:** PSEUDOCODE:

```
public void deleteatbeg()
{
    if(head==null)
    {
        System.out.println("Empty list");
    }
    else if(head.next==null)
    {
        head=null;
    }
    else
    {
        node temp=head.prev;
        temp.next=head.next;
        head=head.next;
        head.prev=temp;
    }
}
```

# **3.3.2. Deletion from the last:** PSEUDOCODE:

```
public void deleteatlast()
    {
        if(head==null)
        {
            System.out.println("Empty list");
        }
        else if(head.next==null)
        {
            head=null;
        }
        else
        {
            node temp=head.prev.prev;
            temp.next=head;
            head.prev=temp;
        }
    }
}
```

# **3.3.3.** Deletion from the specified position:

#### **PSEUDOCODE:**

```
public void deleteatpos(int pos)
     if(pos<1)
                                                       else
       System.out.println("pos<1 is
invalid");
                                                          node temp=head;
                                                          for(int i=1; i < pos-
     else\ if(pos==1)
                                                  1 \& \& temp.next! = head; i++)
       if(head = = null)
                                                            temp=temp.next;
          System.out.println("Empty list");
                                                          if(temp.next!=head)
       else if(head.next==null)
                                                            temp.next=temp.next.next;
                                                            temp.prev=temp;
          head=null;
                                                          else
       else
                                                            System.out.println("invalid
          node temp=head.prev;
                                                  position");
          temp.next=head.next;
          head=head.next;
          head.prev=temp;
```

## SOURCECODE FOR DELETION OPERATION:

```
import java.util.Scanner;
                                                       else
class cdlldelete
                                                         node temp=head.prev;
                                                         temp.next=head.next;
  class node
                                                         head=head.next;
    int data;
                                                         head.prev=temp;
    node next;
    node prev;
    node(int data)
                                                    public void deleteatlast()
       this.data=data;
       this.next=null;
                                                       if(head = = null)
       this.prev=null;
                                                         System.out.println("Empty list");
  public node head=null;
                                                       else if(head.next==null)
                                                         head=null:
  public void add(int x)
    node\ newnode=new\ node(x);
                                                       else
    if(head = = null)
                                                         node temp=head.prev.prev;
       head=newnode;
                                                         temp.next=head;
       head.prev=head;
                                                         head.prev=temp;
       head.next=head;
    }
    else
       node temp=head.prev;
       temp.next=newnode;
                                                    public void deleteatpos(int pos)
       newnode.next=head;
       newnode.prev=temp;
                                                       if(pos<1)
       head.prev=newnode;
       head=newnode;
                                                         System.out.println("pos<1 is
                                                  invalid");
                                                       else\ if(pos==1)
  public void deleteatbeg()
                                                         if(head = = null)
    if(head = = null)
                                                           System.out.println("Empty list");
       System.out.println("Empty list");
                                                         else if(head.next == null)
    else if(head.next==null)
                                                           head=null;
       head=null;
                                                         else
```

```
node temp=head.prev;
                                                    public int count()
         temp.next=head.next;
         head=head.next;
                                                       int c=0;
                                                       if(head==null)
         head.prev=temp;
                                                         return c;
     else
                                                       else
       node temp=head;
                                                         node temp=head;
       for(int i=1; i < pos-
                                                         while(temp.next!=head)
1\&\&temp.next!=head;i++)
                                                           c=c+1;
         temp=temp.next;
                                                           temp=temp.next;
       if(temp.next!=head)
                                                         return c+1;
         temp.next=temp.next.next;
         temp.prev=temp;
                                                    public static void main(String args[])
       else
                                                       cdlldelete d=new cdlldelete();
         System.out.println("invalid
                                                       Scanner input=new
                                                  Scanner(System.in);
position");
                                                      d.add(1);
                                                       d.add(2);
                                                       d.add(3);
                                                       d.add(4);
                                                       d.add(5);
  public void display()
                                                       d.add(6);
                                                       System.out.println("This is circular
                                                  doubly linked list:");
     if(head = = null)
                                                       d.display();
       System.out.println("empty list");
                                                       int ch;
                                                       do
     else
                                                         System.out.println();
       node temp=head;
                                                         System.out.println("From where do
                                                 you want to delete your newnode?");
       while(temp.next!=head)
                                                         System.out.println("From the
         System.out.print(temp.data+"<--
                                                  beginning? Press 1");
>");
                                                         System.out.println("From the last?
                                                  Press 2");
         temp=temp.next;
                                                         System.out.println("From the
                                                  specified position? Press 3");
       System.out.print(temp.data);
                                                         ch=input.nextInt();
                                                         switch(ch)
    node temp=head;
                                                           case 1:
```

```
System.out.println();
            System.out.println("Deleteing
                                                                System.out.println("Deletion
a node from the beginning.");
                                                   at position 2");
            d.deleteatbeg();
                                                                d.deleteatpos(2);
            d.display();
                                                                d.display();
                                                                System.out.println();
            break;
                                                                System.out.println("Deletion
          case 2:
                                                   at position 10");
                                                                d.deleteatpos(10);
            System.out.println("Delete a
                                                                d.display();
node at last.");
                                                                break;
            d.deleteatlast();
            d.display();
                                                              default:
            break;
                                                                System.out.println("Enter
          ļ
                                                   your choice from 1 to 3.");
          case 3:
            System.out.println("Deletion
at pos<1");
                                                        }while(ch<=3);
            d.deleteatpos(-1);
            d.display();
            System.out.println();
            System.out.println("Deletion
at pos=1");
            d.deleteatpos(1);
            d.display();
```

```
This is circular doubly linked list:

6<-->5<-->4<-->3<-->2<-->1

From where do you want to delete your newnode?

From the beginning? Press 1

From the last? Press 2

From the specified position? Press 3

1

Deleting a node from the beginning.

5<-->4<-->3<-->2<-->1

From where do you want to delete your newnode?

From the beginning? Press 1

From the last? Press 2

From the specified position? Press 3

2
```

```
Delete a node at last.
5<-->4<-->3<-->2
From where do you want to delete your newnode?
From the beginning? Press 1
From the last? Press 2
From the specified position? Press 3
Deletion at pos<1
pos<1 is invalid
5<-->4<-->3<-->2
Deletion at pos=1
4<-->3<-->2
Deletion at position 2
4<-->2
Deletion at position 10
invalid position
4<-->2
From where do you want to delete your newnode?
From the beginning? Press 1
From the last? Press 2
From the specified position? Press 3
Enter your choice from 1 to 3.
```

### **3.4. PROGRAM 4:** Count the element of the linked list.

## PSEUDOCODE:

### SOURCECODE:

```
class cdllcount
                                                        head=newnode;
  class node
    int data;
                                                   public void display()
    node next;
    node prev;
                                                      if(head = = null)
    node(int data)
                                                        System.out.println("empty list");
       this.data=data;
       this.next=null;
                                                     else
       this.prev=null;
                                                        node temp=head;
                                                        while(temp.next!=head)
  public node head=null;
                                                          System.out.print(temp.data+"<--
                                                 >");
  public void add(int x)
                                                          temp=temp.next;
    node\ newnode=new\ node(x);
    if(head = = null)
                                                        System.out.print(temp.data);
       head=newnode;
       head.prev=head;
                                                      node temp=head;
       head.next=head;
    }
    else
                                                   public int count()
       node temp=head.prev;
                                                      int c=0;
       temp.next=newnode;
                                                      if(head==null)
       newnode.next=head;
       newnode.prev=temp;
                                                        return c;
       head.prev=newnode;
```

```
else
                                                  {
                                                     cdllcount s=new cdllcount();
     node temp=head:
                                                     System.out.println("A circular doubly
     while(temp.next!=head)
                                                linked list is given as:");
                                                     s.add(5);
                                                     s.add(10);
       c=c+1;
                                                     s.add(15);
       temp=temp.next;
                                                     s.display();
                                                     System.out.println();
     return c+1;
                                                     System.out.println("The number of
                                                nodes in the linked list are: "+s.count());
public static void main(String args[])
```

```
A circular doubly linked list is given as:
15<-->10<-->5
The number of nodes in the linked list are:3
```

## 4. OUTPUT AND DISCUSSION:

- ✓ In this experiment, the output of program 1 shows the traverse of a node to display the data of the node in the circular doubly linked list.
- ✓ In the output of program 2, it can be seen that the insertion of a node is done at the beginning of the linked list, after then, when the user press 2, a node is inserted at the last and then when the user press option 3, a node is inserted at a specified position.
- ✓ The output of the Program 3 shows the deletion of a node from the beginning, from the last and from the specified position.
- ✓ In program 4, the number of nodes are counted in the given circular doubly linked list..

# 5. CONCLUSION:

A Java program to traverse, insert a node at first, last, certain position, delete a node at beginning, at last and at certain position and counting of the nodes in circular doubly linked list is successfully run.