

DATA STRUCTURES (017013292)
Semester – II
Chapter Name: DOUBLY AND CIRCULAR LINK LIST

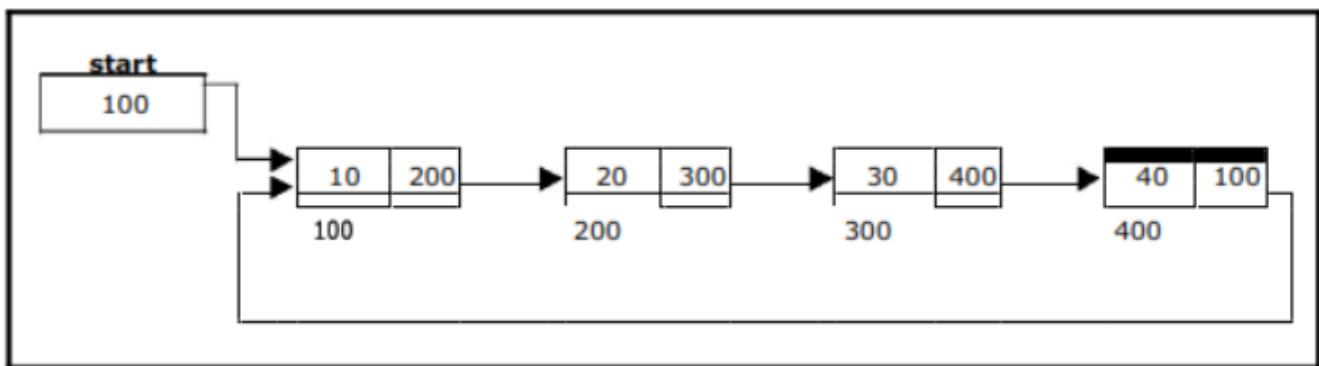
Contents

| | |
|--|----|
| 1. Circular Linked List..... | 2 |
| 2. Advantages of Circular Linked Lists: | 2 |
| 4. Creating a Node of Circular LinkedList in JAVA | 3 |
| 5. Method to Insert a New Node (n) at the Beginning of Circular LinkedList | 4 |
| 6. Method to Insert a New Node (n) at Last of Circular LinkedList..... | 5 |
| 7. Method to Insert a New Node (n) Before A Particular Value in a Circular LinkedList | 6 |
| 8. Method to Delete the First Node in Circular LinkedList..... | 7 |
| 9. Method to Delete the Last Node in Circular LinkedList | 8 |
| 10. Display Method for Circular LinkedList..... | 9 |
| 11. Java Program to InsertAtFirst, InsertAtLast, DeleteFirst & DeleteLast in Circular LinkedList | 10 |

DATA STRUCTURES (017013292)
Semester – II
Chapter Name: DOUBLY AND CIRCULAR LINK LIST

1. Circular Linked List

- ◆ Circular linked list is a linked list where all nodes are connected to form a circle. There is no NULL at the end. A circular linked list can be a singly circular linked list or doubly circular linked list.
- ◆ Circular linked lists are frequently used instead of ordinary linked list because many operations are much easier to implement. In circular linked list no null pointers are used, hence all pointers contain valid address.



2. Advantages of Circular Linked Lists:

1. Any node can be a starting point. We can traverse the whole list by starting from any point. We just need to stop when the first visited node is visited again.
2. Useful for implementation of queue. Unlike this implementation, we don't need to maintain two pointers for front and rear if we use circular linked list. We can maintain a pointer to the last inserted node and front can always be obtained as next of last.
3. Circular lists are useful in applications to repeatedly go around the list. For example, when multiple applications are running on a PC, it is common for the operating system to put the running applications on a list and then to cycle through them, giving each of them a slice of time to execute, and then making them wait while the CPU is given to another application. It is convenient for the operating

4. Creating a Node of Circular LinkedList in JAVA

```
42 | public static class CircularLinkedList //created a class named Circular LL//  
43 | {  
44 |     class Node //created a node which contains a data and refrence//  
45 |     {  
46 |         int data; //the node will have a data of integer type//  
47 |         Node next; //it will have a reference variable//  
48 |  
49 |         Node(int data) //created a constructor to create a node//  
50 |         {  
51 |             this.data = data; //the value of data will be same as entered by user//  
52 |             this.next = null; //the next pointer of the node will initialized with null//  
53 |         }  
54 |     }  
55 | }
```

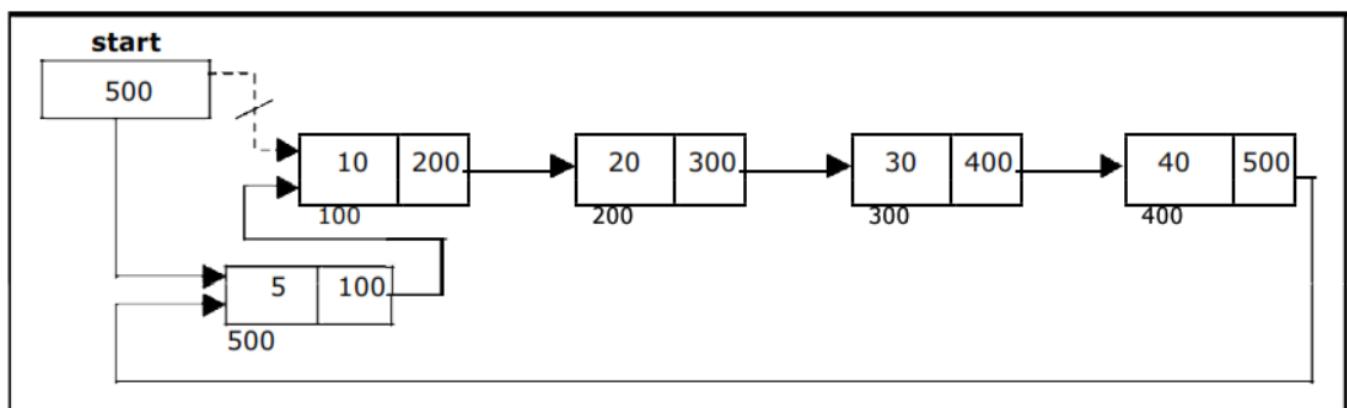
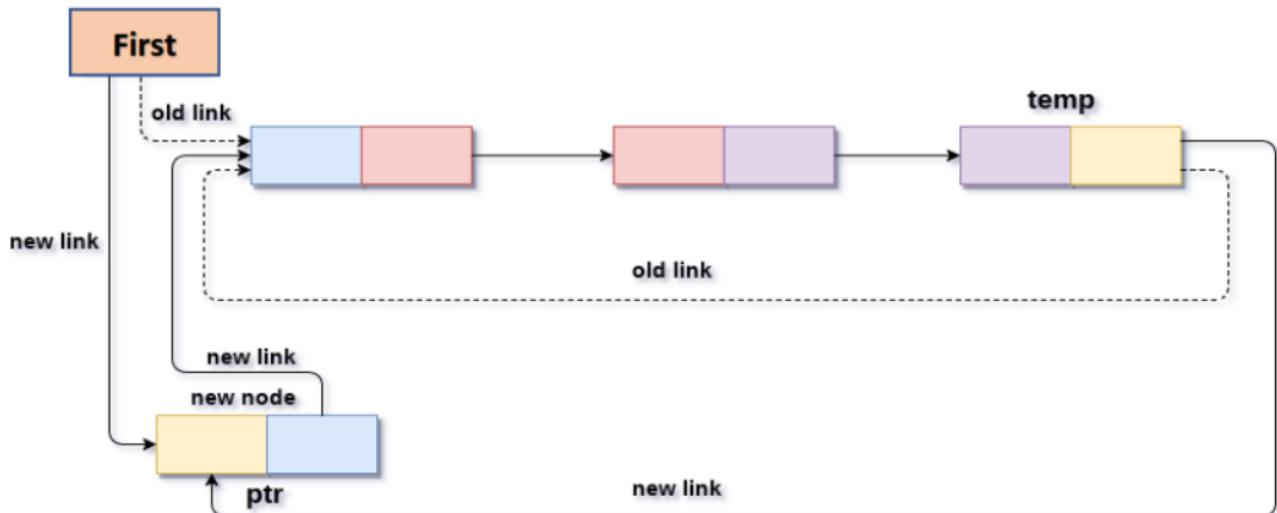
DATA STRUCTURES (017013292)
Semester – II
Chapter Name: DOUBLY AND CIRCULAR LINK LIST

5. Method to Insert a New Node (n) at the Beginning of Circular LinkedList

```

56
57
58     void insertFirst(int data) //created a method to insert a new node at start//
59
60     {
61         Node n = new Node(data); //In insert method we always required to create a node//
62         if (first == null) //checking if LinkedList is empty or not//
63         {
64             first = n; //if yes then first shall be directly pointing to n//
65             n.next = n; //and new node's next should be pointing n due to CLL//
66         }
67         else //if LinkedList has more values//
68         {
69             n.next = first; //new node n's next shall be equal to first//
70             Node temp = first; //created a temp node with the same value as first//
71             while (temp.next != first) //until we get the temp = null this loop will run//
72                 temp = temp.next;
73             temp.next = n; //now temp's next which is showing old first shall be pointing n//
74             first = n; //and now first shall be poing towards n//
75         }
76         System.out.println(data+" is inserted First");
77     }

```

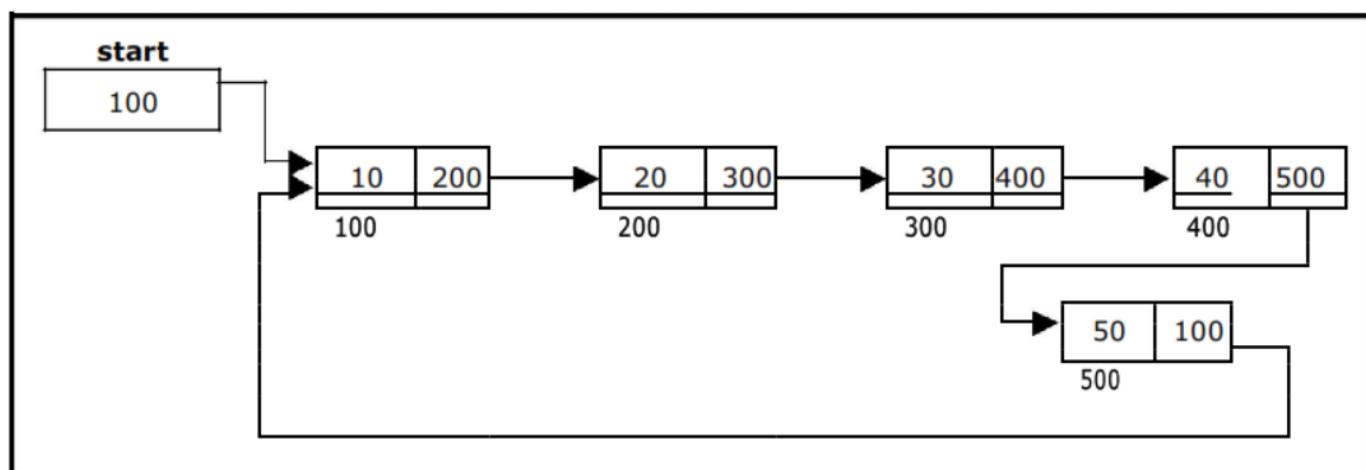
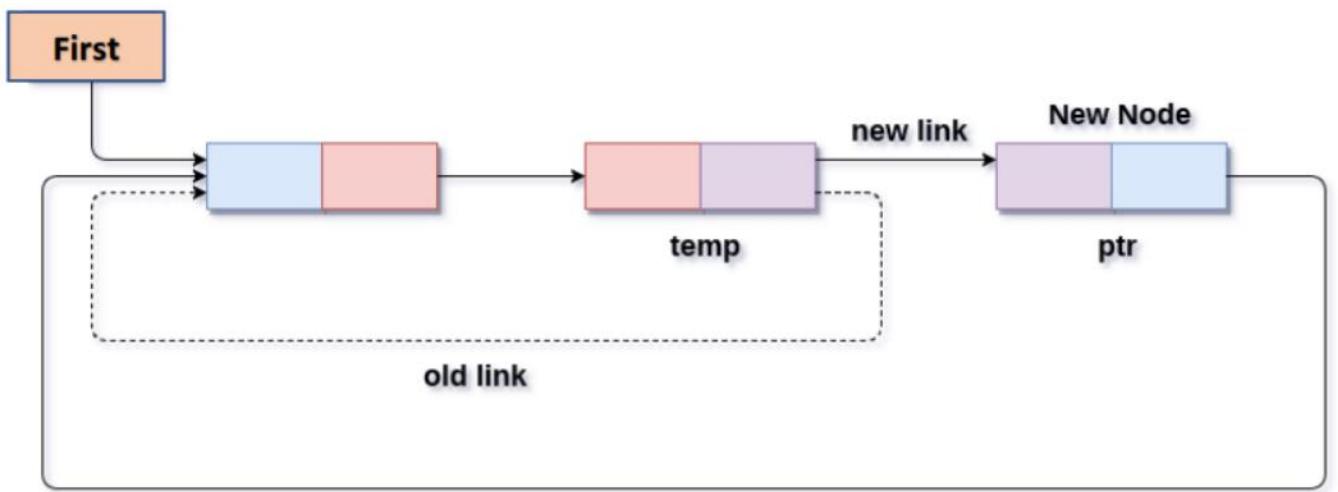


6. Method to Insert a New Node (n) at Last of Circular LinkedList

```

75 |
76 |-----|
77 |-----|
78 |-----|
79 |-----|
80 |-----|
81 |-----|
82 |-----|
83 |-----|
84 |-----|
85 |-----|
86 |-----|
87 |-----|
88 |-----|
89 |-----|
90 |-----|
91 |-----|
92 |-----|
    void insertlast(int data) //created a method to Insert a Node at Last//
    {
        Node n = new Node(data); //In insert method we always required to create a node//
        if (first == null) //checking if LinkedList is empty or not//
        {
            first = n; //if yes then first shall be directly pointing to n//
            n.next = n; //and new node's next should be pointing n due to CLL//
        }
        else //otherwise//
        {
            Node temp = first; //created a temp node with the same value as first//
            while (temp.next != first) //until we get the temp = null this loop will run//
                temp = temp.next;
            temp.next = n; //now temp's next which is showing old first shall be pointing n//
            n.next = first; //new node n's next shall be equal to first//
        }
        System.out.println(data+" is inserted Last");
    }

```



7. Method to Insert a New Node (n) Before A Particular Value in a Circular LinkedList

```

void insertValuebefore(int data,int target) //created a method insert a new node before values//
{
    Node new_node=new Node(data); //In insert method we always required to create a node//
    int flag=0; //used a flag variable to check the value exists in LL or not//
    if(first==null) //checking that LinkedList is empty or not//
    {
        System.out.println("Empty Circular Linked List, So cant insert value before given value ");
    }
    else //otherwise//
    {
        Node temp=first; //created a temp node with the same value as first//
        do
        {
            if(temp.data==target) //if we get temp's data equal to target//
            {
                flag=1; //the flag value will become 1//
            }
            temp=temp.next;
        }while(temp!=first); //untill temp again becomes first in CLL//
        if (flag==1) //now if flag is 1 means value matching//
        {
            if (first.data==target && first.next==first) //checking for 1 node condition//
            {
                new_node.next=first; //new node's next shall be point to first//
                first.next=new_node; //first's next shall be new node//
                first=new_node; //first now can point to new node//
            }
            else if (first.data==target && first.next!=first) //otherwise//
            {
                new_node.next=first; //new node's next eual to first//
                while(temp.next!=first) //until we get the temp's next = null this loop will run//
                {
                    temp=temp.next;
                }
                temp.next=new_node; //temp's next shall be equal to new node//
                first=new_node; //first shall now point to new node//
            }
            else
            {
                //Node temp=first;
                while(temp.next.data!=target)//until we get temp's next's data will be equal to target//
                {
                    temp=temp.next;
                }
                new_node.next=temp.next; // new node's next equal to temp's next//
                temp.next=new_node; //temp's next will be equal to new node//
            }
        }
        System.out.println("Target Value does not exist");
    }
}

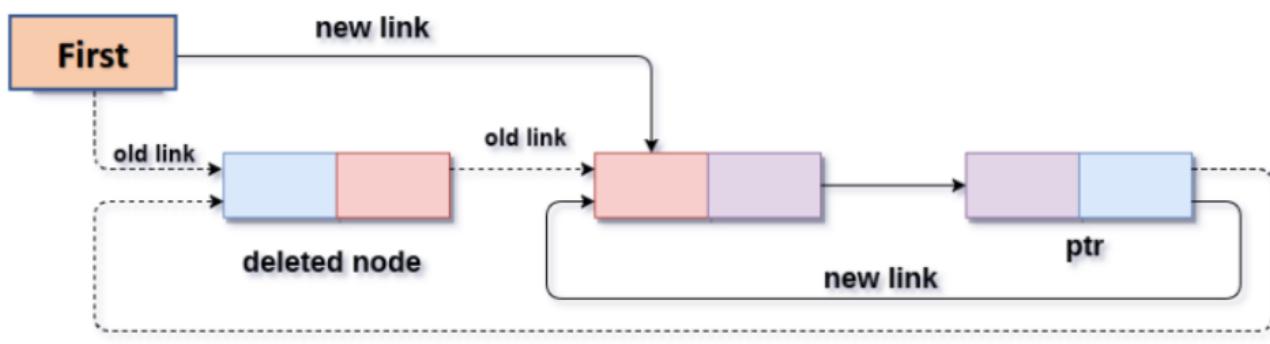
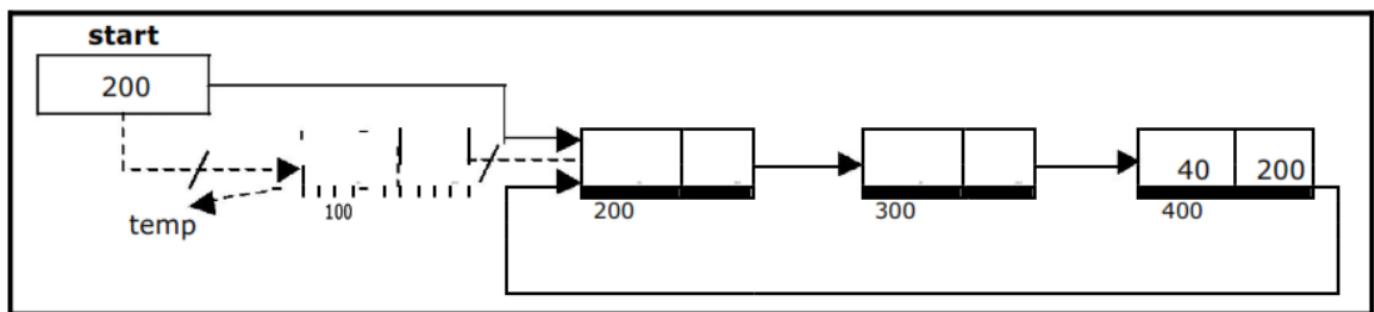
```

8. Method to Delete the First Node in Circular LinkedList

```

69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
    void deleteFirst() //created a method to Delete a Node at Last//
    {
        if(first==null) //checking if LinkedList is empty or not//
        {
            System.out.println("Empty");
        }
        else if(first.next==null && first.prev==null) //checking if the LinkedList has only 1 node//
        {
            first=null; //if yes then make first equal to null//
        }
        else //if linked list has more elements then//
        {
            Node temp=first; //created a temp node with the same value as first//
            first=first.next; //shifted the first pointer to first's next//
            first.prev=null; //make first's prev pointer null//
            temp.next=null; //make temp(which is at first)'s next pointer null//
        }
    }

```



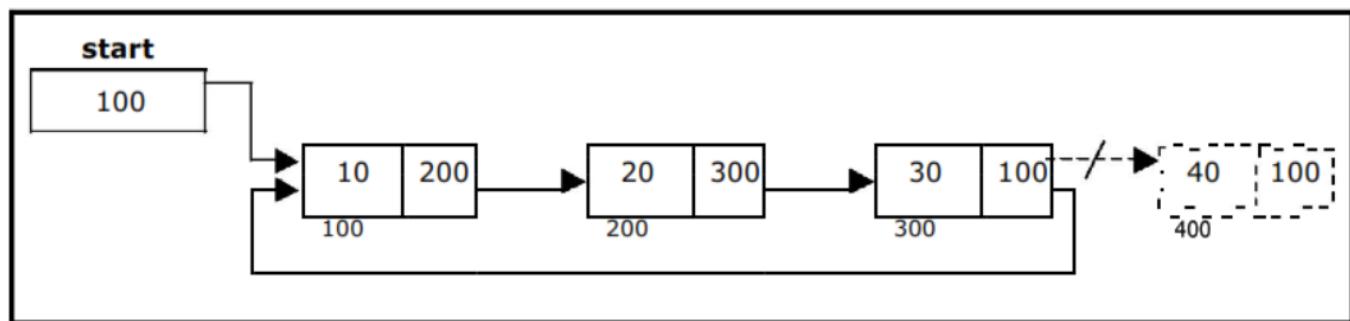
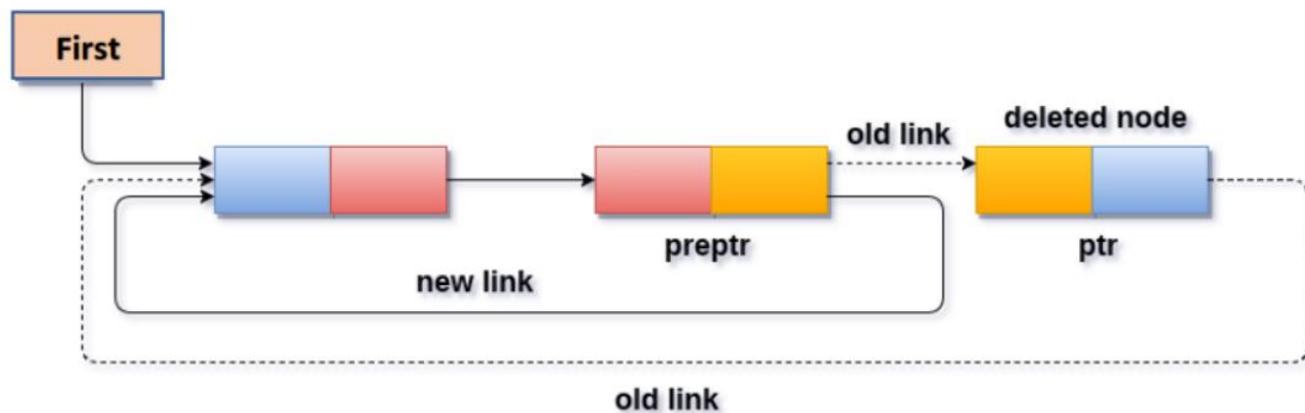
9. Method to Delete the Last Node in Circular LinkedList

```

117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139

    void dellast() //created a method to Delete a Node at Last//
    {
        if (first == null) //checking if LinkedList is empty or not//
        {
            System.out.println("UnderFlow");
        }
        else if(first.next==first) //checking if LL has only 1 node//
        {
            first=null; //then first shall be null//
        }
        else
        {
            Node temp = first; //created a temp node with the same value as first//
            while (temp.next != first) //until we get the temp's next's next = null this loop will run/
            {
                temp = temp.next;
            }
            Node del=temp.next;//to release memory of deleted node
            temp.next = first; //temp's next shall be equal to first//
            System.out.println(del.data+" is deleted from last");
            del.next=null;
        }
    }
}

```



10. Display Method for Circular LinkedList

```
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256
```

```
void display() //created a method to display the data of CircularLinkedList//  
{  
    if(first == null) //checking if LinkedList is empty or not//  
    {  
        System.out.println("Empty");  
    }  
    else  
    {  
        Node temp = first; //created a temp node with the same value as first//  
        while (temp.next != first) //until we get the temp's next = null this loop will run//  
        {  
            System.out.print(temp.data + "-"); //temp's data get printed//  
            temp = temp.next;  
        }  
        System.out.println(temp.data + " : Circular Linked List"); //printing last node value//  
    }  
}
```

11. Java Program to InsertAtFirst, InsertAtLast, DeleteFirst & DeleteLast in Circular LinkedList

```

1  class Run7
2  {
3      public static void main(String args[]) {
4          CircularLinkedList L = new CircularLinkedList();
5          L.insertlast(10);
6          L.display();
7          L.insertlast(20);
8          L.display();
9          L.delTargetvalue(20);
10         L.display();
11         L.insertValuebefore(10,10);
12         L.display();
13         L.insertFirst(10);
14         L.display();
15         L.insertFirst(20);
16         L.display();
17         L.insertFirst(30);
18         L.display();
19         L.insertValuebefore(111,20);
20         L.display();
21         L.insertFirst(1);
22         L.display();
23         L.insertFirst(2);
24         L.display();
25         L.insertFirst(3);
26         L.display();
27         L.insertlast(10);
28         L.display();
29         L.insertlast(20);
30         L.display();
31         L.insertlast(30);
32         L.display();
33         L.insertValuebefore(111,30);
34         L.display();
35         L.delTargetvalue(111);
36         L.display();
37         L.dellast();
38         L.display();
39         L.delfirst();
40         L.display();
41     }
42     public static class CircularLinkedList //created a class named Circular LL//
43     {
44         class Node //created a node which contains a data and refrence//
45         {
46             int data; //the node will have a data of integer type//
47             Node next; //it will have a reference variable//
48
49             Node(int data) //created a constructor to create a node//
50             {
51                 this.data = data; //the value of data will be same as entered by user//
52                 this.next = null; //the next pointer of the node will initialized with
53                 null////
54             }
55         }
56         Node first = null;
57         void insertFirst(int data) //created a method to insert a new node at start//
58         {
59             Node n = new Node(data); //In insert method we always required to create a
60             //node//
61             if (first == null) //checking if LinkedList is empty or not//
62             {
63                 first = n; //if yes then first shall be directly pointing to n//
64                 n.next = n; //and new node's next should be pointing n due to CLL//
65             }
66             else //if LinkedList has more values//
67             {

```

DATA STRUCTURES (017013292)
Semester – II
Chapter Name: DOUBLY AND CIRCULAR LINK LIST

```
66         n.next = first; //new node n's next shall be equal to first//  
67         Node temp = first; //created a temp node with the same value as first//  
68         while (temp.next != first) //until we get the temp's next = null this  
69             loop will run//  
70             temp = temp.next;  
71             temp.next = n; //now temp's next which is showing old first shall be  
72             pointing n//  
73             first = n; //and now first shall be poing towards n//  
74         }  
75         System.out.println(data+" is inserted First");  
76     }  
77     void insertlast(int data) //created a method to Insert a Node at Last//  
78     {  
79         Node n = new Node(data); //In insert method we always required to create a  
80         node//  
81         if (first == null) //checking if LinkedList is empty or not//  
82         {  
83             first = n; //if yes then first shall be directly pointing to n//  
84             n.next = n; //and new node's next should be pointing n due to CLL//  
85         }  
86         else //otherwise//  
87         {  
88             Node temp = first; //created a temp node with the same value as first//  
89             while (temp.next != first) //until we get the temp's next = null this  
90             loop will run//  
91             temp = temp.next;  
92             temp.next = n; //now temp's next which is showing old first shall be  
93             pointing n//  
94             n.next = first; //new node n's next shall be equal to first//  
95         }  
96         System.out.println(data+" is inserted Last");  
97     }  
98     void delfirst() //created a method to Delete a Node at Last//  
99     {  
100         if (first == null) //checking if LinkedList is empty or not//  
101         {  
102             System.out.println("UnderFlow");  
103         }  
104         else if(first.next==first) //checking if LL has only 1 node//  
105         {  
106             first=null; //then first shall be null//  
107         }  
108         else //otherwise//  
109         {  
110             Node del = first;//to release memory of deleted node  
111             Node temp = first; //created a temp node with the same value as first//  
112             while (temp.next != first) //until we get the temp's next = null this  
113             loop will run//  
114             {  
115                 temp = temp.next;  
116             }  
117             temp.next = first.next; //then temp's next shall be equal to first's  
118             next//  
119             first = first.next; //first shall be equal to first's next//  
120             del.next = null;  
121             System.out.println(del.data+" is deleted from first");  
122         }  
123     }  
124     void dellast() //created a method to Delete a Node at Last//  
125     {  
126         if (first == null) //checking if LinkedList is empty or not//  
127         {  
128             System.out.println("UnderFlow");  
129         }  
130         else if(first.next==first) //checking if LL has only 1 node//  
131         {  
132             first=null; //then first shall be null//  
133         }  
134     }
```

DATA STRUCTURES (017013292)
Semester – II
Chapter Name: DOUBLY AND CIRCULAR LINK LIST

```
126     }
127     else
128     {
129         Node temp = first; //created a temp node with the same value as first//
130         while (temp.next.next != first) //until we get the temp's next's next =
131             null this loop will run//
132         {
133             temp = temp.next;
134         }
135         Node del=temp.next;//to release memory of deleted node
136         temp.next = first; //temp's next shall be equal to first//
137         System.out.println(del.data+" is deleted from last");
138         del.next=null;
139     }
140     void insertValuebefore(int data,int target) //created a method insert a new node
141     before values//
142     {
143         Node new_node=new Node(data); //In insert method we always required to
144             create a node//
145         int flag=0; //used a flag variable to check the value exists in LL or not//
146         if(first==null) //checking that LinkedList is empty or not//
147         {
148             System.out.println("Empty Circular Linked List, So cant insert value
149             before given value ");
150         }
151         else //otherwise//
152         {
153             Node temp=first; //created a temp node with the same value as first//
154             do
155             {
156                 if(temp.data==target) //if we get temp's data equal to target//
157                 {
158                     flag=1; //the flag value will become 1//
159                 }
160                 temp=temp.next;
161             }while(temp!=first); //untill temp again becomes first in CLL//
162             if (flag==1) //now if flag is 1 means value matching//
163             {
164                 if (first.data==target && first.next==first) //checking for 1 node
165                 condition//
166                 {
167                     new_node.next=first; //new node's next shall be point to first//
168                     first.next=new_node; //first's next shall be new node//
169                     first=new_node; //first now can point to new node//
170                 }
171                 else if (first.data==target && first.next!=first) //otherwise//
172                 {
173                     new_node.next=first; //new node's next equal to first//
174                     while(temp.next!=first) //until we get the temp's next = null
175                         this loop will run//
176                     {
177                         temp=temp.next;
178                     }
179                     temp.next=new_node; //temp's next shall be equal to new node//
180                     first=new_node; //first shall now point to new node//
181                 }
182                 else
183                 {
184                     //Node temp=first;
185                     while(temp.next.data!=target)//until we get temp's next's data
186                     will be equal to target//
187                     {
188                         temp=temp.next;
189                     }
190                     new_node.next=temp.next; // new node's next equal to temp's
191                     next//
```

DATA STRUCTURES (017013292)
Semester – II
Chapter Name: DOUBLY AND CIRCULAR LINK LIST

```
185                     temp.next=new_node; //temp's next will be equal to new node//
186                 }
187             }
188             System.out.println("Target Value does not exist");
189         }
190     }
191     void delTargetvalue(int target) //created a method delete a value//
192     {
193         int flag=0; //used a flag variable to check the value exists in LL or not//
194         if(first==null) //checking that LinkedList is empty or not//
195         {
196             System.out.println("Empty Circular Linked List, So cant insert value
197             before given value ");
198         }
199         else //otherwise//
200         {
201             Node temp=first; //created a temp node with the same value as first//
202             do
203             {
204                 if(temp.data==target) //if we get temp's data equal to target//
205                 {
206                     flag=1; //the flag value will become 1//
207                 }
208                 temp=temp.next;
209             }while(temp!=first); //untill temp again becomes first in CLL//
210             if (flag==1) //now if flag is 1 means value matching//
211             {
212                 if (first.data==target && first.next==first) //checking for 1 node
213                 condition//
214                 {
215                     first=null;
216                 }
217                 else if (first.data==target && first.next!=first) //otherwise//
218                 {
219                     Node del=first; //created a new node del equal to first//
220                     while(temp.next!=first) //until we get the temp's next = null
221                     this loop will run//
222                     {
223                         temp=temp.next;
224                     }
225                     temp.next=first.next; //temp's next equal to first's next//
226                     first=first.next; //first shall be now equal to first's next//
227                     del.next=null; //del's next becomes null to break link//
228                 }
229                 else
230                 {
231                     //Node temp=first;
232                     while(temp.next.data!=target) //until we get temp's next's data
233                     will be equal to target//
234                     {
235                         temp=temp.next;
236                     }
237                     temp.next=temp.next.next; //temp's next shall be equal to temp's
238                     next's next//}
239                 }
240             }
241             System.out.println("Target Value does not exist");
242         }
243     void display() //created a method to display the data of CircularLinkedList//
244     {
245         if(first == null) //checking if LinkedList is empty or not//
246         {
247             System.out.println("Empty");
248         }
249         else
250         {
```

DATA STRUCTURES (017013292)
Semester – II
Chapter Name: DOUBLY AND CIRCULAR LINK LIST

```
247     Node temp = first; //created a temp node with the same value as first//  
248     while (temp.next != first) //until we get the temp's next = null this  
249         loop will run//  
250     {  
251         System.out.print(temp.data + "-"); //temp's data get printed//  
252         temp = temp.next;  
253     }  
254     System.out.println(temp.data + " : Circular Linked List"); //printing  
255         last node value//  
256     }  
257 }
```