

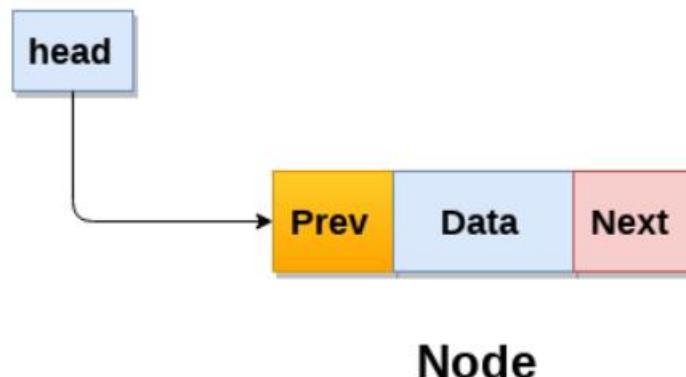
DATA STRUCTURES (017013292)
Semester – II
Chapter Name: DOUBLY AND CIRCULAR LINK LIST

Contents

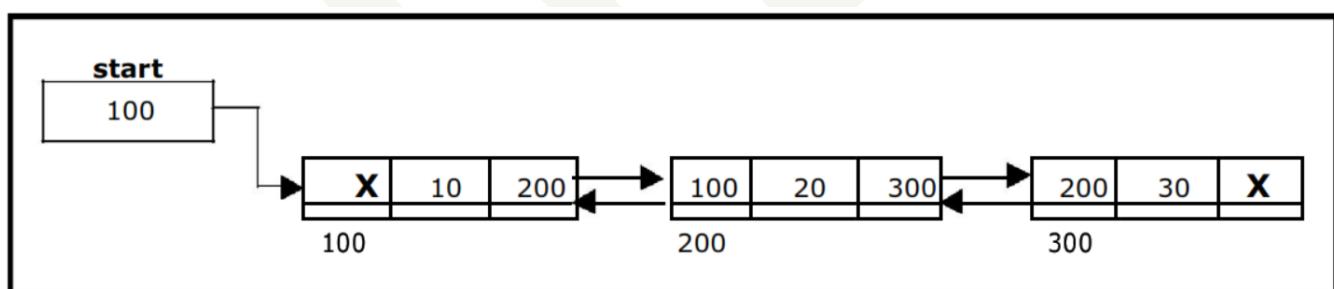
1. Doubly Linked List	2
2. Creating a Node of Doubly LinkedList in JAVA	3
3. Advantages of Doubly LinkedList over Singly LinkedList	3
4. Disadvantages of Doubly LinkedList over Singly LinkedList.....	3
5. Method to Insert a New Node (n) at the Beginning of Doubly LinkedList	4
6. Method to Insert a New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

1. Doubly Linked List

- ◆ Doubly linked list is a complex type of linked list in which a node contains a pointer to the previous as well as the next node in the sequence.
- ◆ Therefore, in a doubly linked list, a node consists of three parts: node data, pointer to the next node in sequence (next pointer), pointer to the previous node (previous pointer). A sample node in a doubly linked list is shown in the figure.



A doubly linked list containing three nodes having numbers having number 10,20 & 30 in their data part, is shown in the following image.



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

2. Creating a Node of Doubly LinkedList in JAVA

```
2 class DLL //created a class name DLL//  
3 {  
4     class Node //Created a nested class Node//  
5     {  
6         int data; //A node is having data of integer type//  
7         Node next; //A node is having next reference//  
8         Node prev; //A node is having prev reference//  
9         Node (int data) //created a constructor with parameter//  
10        {  
11            this.data=data; //data will the same value entered by user//  
12            next=null; //next reference will be initially null//  
13            prev=null; //prev reference will be initially null//  
14        }  
15    }  
}
```

3. Advantages of Doubly LinkedList over Singly LinkedList

1. A DLL can be traversed in both forward and backward direction.
2. The delete operation in DLL is more efficient if pointer to the node to be deleted is given.
3. We can quickly insert a new node before a given node.

In singly linked list, to delete a node, pointer to the previous node is needed. To get this previous node, sometimes the list is traversed. In DLL, we can get the previous node using previous pointer.

4. Disadvantages of Doubly LinkedList over Singly LinkedList

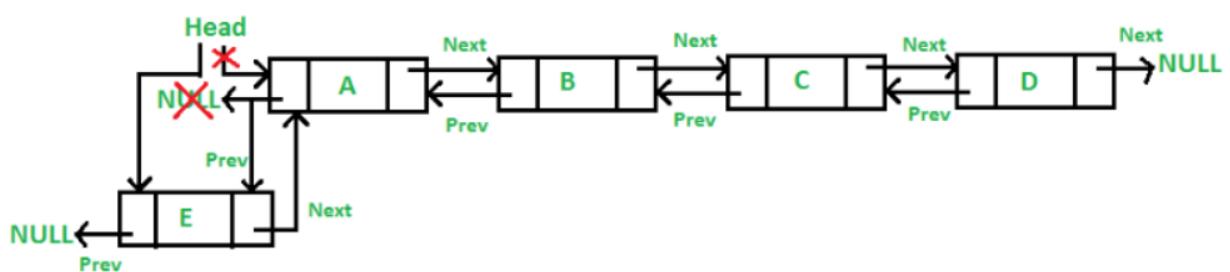
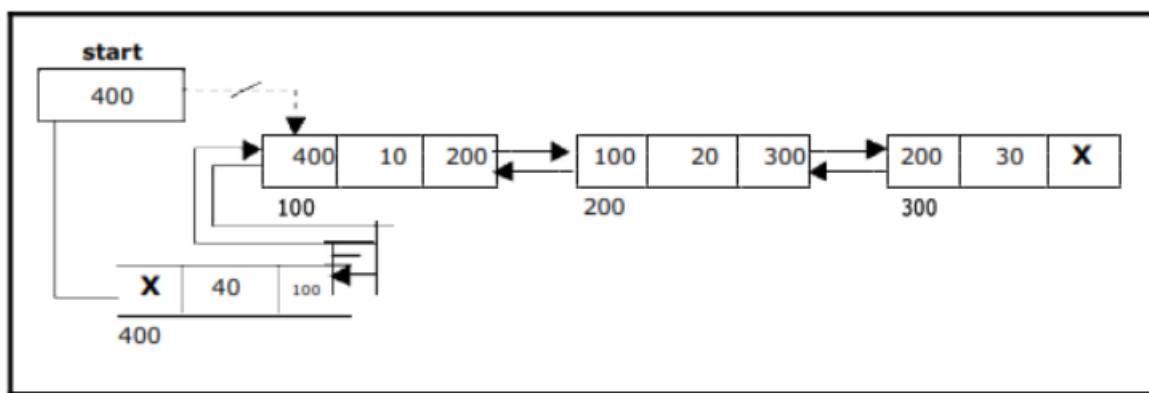
1. Every node of DLL Require extra space for a previous pointer. It is possible to implement DLL with single pointer.
2. All operations require an extra pointer previous to be maintained. For example, in insertion, we need to modify previous pointers together with next pointers. For example in following functions for insertions at different positions, we need 1 or 2 extra steps to set previous pointer.

Node = Prev + Data + Next

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

```

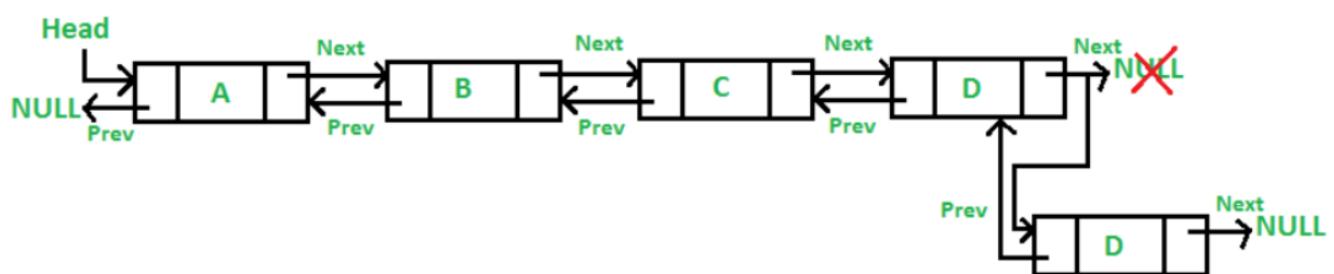
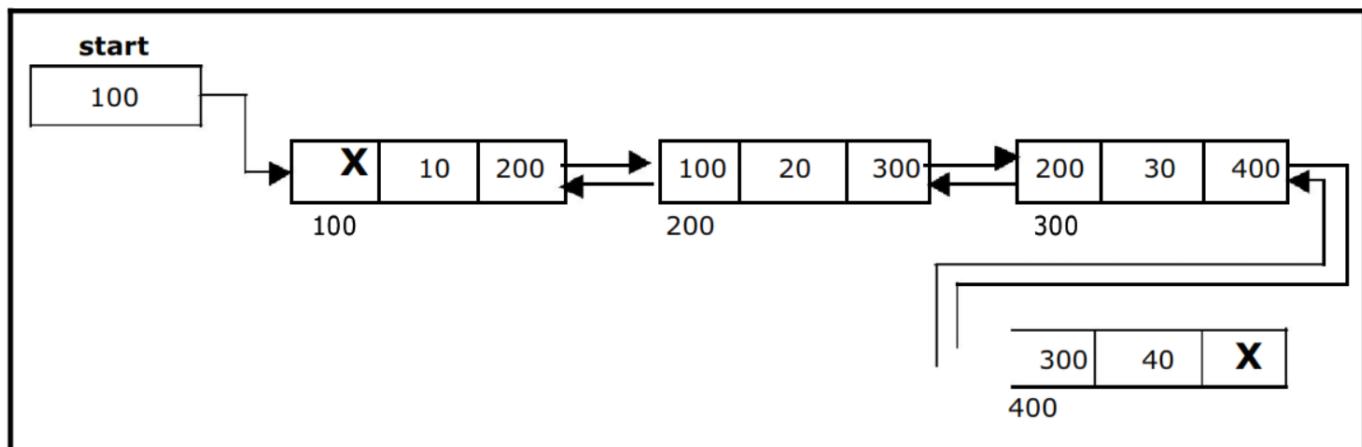
35 void insertAtFirst(int data) //created a method to Insert a Node at First//
36 {
37     Node n= new Node(data); //In insert method we always required to create a node//
38     if(first == null) //checking if LinkedList is empty or not//
39     {
40         first =n; //If empty then directly value of n will be assigned to first//
41     }
42     else //If LinkedList has more values then//
43     {
44         n.next=first; //the new node n's next will be set to the first//
45         first.prev=n; //first node's previous will be set to new node n//
46         first=n; //now the first shall point at n//
47     }
48 }
```



6. Method to Insert a New Node (n) at Last of Doubly Linked List

```

49
50
51 void insertAtLast(int data) //created a method to Insert a Node at Last//
{
52     Node n= new Node(data); //In insert method we always required to create a node//
53     Node temp=first; //created a temp node with the same value as first//
54     if(first == null) //checking if LinkedList is empty or not//
55     {
56         n.prev=null; //In this case new node n's prev shall be null// 
57         first=n; //and the first now shall point to the new node n//
58     }
59     else //otherwise//
60     {
61         while(temp.next!=null) //until we get the temp = null this loop will run//
62         {
63             temp=temp.next;
64         }
65         n.prev=temp; //New node n's prev should be equal temp// 
66         temp.next=n; //And temp's next shall be equal to new node n//
67     }
68 }
```



7. Method to Insert a New Node Before Particular Target Value in Doubly Linked List

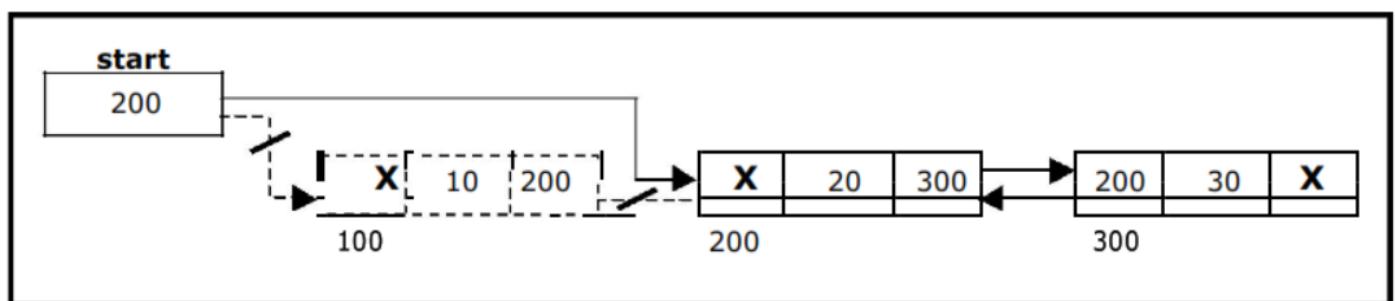
```
109
110
111
112
113 void InsertBeforeValue(int data, int target)
114 {
115     Node newNode = new Node(data); // Create a new node with the given data
116
117     if (first == null) { // Check if the list is empty
118         System.out.println("LinkedList is Empty");
119         return;
120     }
121
122     if (first.data == target)
123     { // Check if the first node has the target value
124         newNode.next = first; // Set the new node's next to the first node
125         first.prev = newNode; // Set the first node's prev to the new node
126         first = newNode; // Update the first node to be the new node
127         return;
128     }
129
130     Node current = first.next; // Start from the second node
131     while (current != null)
132     {
133         if (current.data == target)
134         {
135             newNode.next = current; // Set the new node's next to the current node
136             newNode.prev = current.prev; // Set the new node's prev to the current node's prev
137             current.prev.next = newNode; // Update the previous node's next to the new node
138             current.prev = newNode; // Update the current node's prev to the new node
139             return;
140         }
141         current = current.next; // Move to the next node
142     }
143
144     // If the target value is not found
145     System.out.println("Target value not found in the list");
146 }
```

8. Method to Delete the First Node in Doubly Linked List

```

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//
        }
    }

```

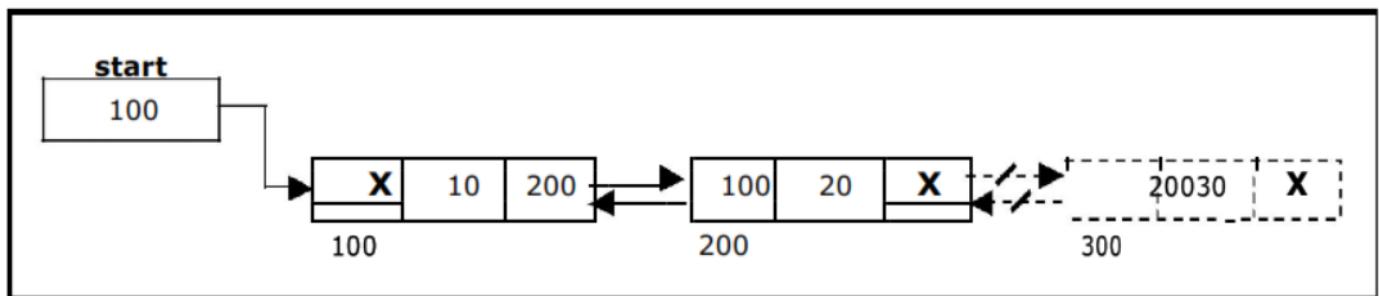


9. Method to Delete the Last Node in Doubly Linked List

```

void deleteLast() //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//
        while(temp.next!=null) //until we get the temp = null this loop will run//
        {
            temp=temp.next; //temp will shift to temp's next//
        }
        temp.prev.next=null; //temp's prev's next will be null// 
        temp.prev=null; //and then temp's prev will be also null//
    }
}

```



10.**Method to Delete a Particular Value in Doubly LinkedList**

```
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180

void deleteValue(int value)
{
    if (first == null)
    {
        System.out.println("LINKLIST IS EMPTY");
    }
    else if (first.data == value)
    {
        System.out.println(first.data);
        first = first.next;
        first.prev = null;
    }
    else
    {
        Node temp = first;
        while (temp.next != null && temp.data != value) {
            temp = temp.next;
        }
        if (temp.data == value)
        {
            if (temp.next != null)
            {
                temp.prev.next = temp.next;
                temp.next.prev = temp.prev;
            }
            else
            {
                temp.prev.next = null;
                temp.prev = null;
            }
        }
        else
        {
            System.out.println("VALUE NOT FOUND");
        }
    }
}
```



11. Display Method for Doubly LinkedList

```
17 void displayD() //created a method to display the data of DoublyLinkedList//  
18 {  
19     System.out.print("NULL-"); //As DLL starts with NULL, this will always print NULL at Start//  
20     if(first==null) //Checking if LinkedList is Empty or not//  
21     {  
22         System.out.println("LinkedList is Empty");  
23     }  
24     else  
25     {  
26         Node temp = first; //created a temp node with the same value as first//  
27         while(temp!=null) //until we get the temp = null this loop will run//  
28         {  
29             System.out.print(temp.data + "-");  
30             temp=temp.next;  
31         }  
32         System.out.println("NULL"); //each DLL ends with NULL//  
33     }  
34 }
```

12. Java Program to InsertAtFirst, InsertAtLast, DeleteFirst & DeleteLast in Doubly LinkedList

```

1 import java.util.*;
2 class DLL //created a class name DLL//
3 {
4     class Node //Created a nested class Node//
5     {
6         int data; //A node is having data of integer type//
7         Node next; //A node is having next reference//
8         Node prev; //A node is having prev reference//
9         Node (int data) //created a constructor with parameter//
10        {
11            this.data=data; //data will the same value entered by user//
12            next=null; //next reference will be initially null//
13            prev=null; //prev reference will be initially null//
14        }
15    }
16    Node first=null;
17    void displayD() //created a method to display the data of DoublyLinkedList//
18    {
19        System.out.print("NULL-"); //As DLL starts with NULL, this will always print
20        //NULL at Start//
21        if(first==null) //Checking if LinkedList is Empty or not//
22        {
23            System.out.println("LinkedList is Empty");
24        }
25        else
26        {
27            Node temp = first; //created a temp node with the same value as first//
28            while(temp!=null) //until we get the temp = null this loop will run//
29            {
30                System.out.print(temp.data + "-");
31                temp=temp.next;
32            }
33            System.out.println("NULL"); //each DLL ends with NULL//
34        }
35    void insertAtFirst(int data) //created a method to Insert a Node at First//
36    {
37        Node n= new Node(data); //In insert method we always required to create a
38        //node//
39        if(first == null) //checking if LinkedList is empty or not//
40        {
41            first =n; //If empty then directly value of n will be assigned to first//
42        }
43        else //If LinkedList has more values then//
44        {
45            n.next=first; //the new node n's next will be set to first//
46            first.prev=n; //first prev will be set to n//
47            first=n; //now the first shall point at n//
48        }
49    void insertAtLast(int data) //created a method to Insert a Node at Last//
50    {
51        Node n= new Node(data); //In insert method we always required to create a
52        //node//
53        Node temp=first; //created a temp node with the same value as first//
54        if(first == null) //checking if LinkedList is empty or not//
55        {
56            n.prev=null; //In this case new node n's prev shall be null//
57            first=n; //and the first now shall point to the new node n//
58        }
59        else //otherwise//
60        {
61            while(temp.next!=null) //until we get the temp = null this loop will
62            //run//
63            {
64                temp=temp.next;

```

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

```
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127

        }
        n.prev=temp; //New node n's prev should be equal temp//
        temp.next=n; //And temp's next shall be equal to new node n//
    }
}
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//
    }
}
void deleteLast() //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//
        while(temp.next!=null) //until we get the temp = null this loop will
run//
        {
            temp=temp.next; //temp will shift to temp's next//
        }
        temp.prev.next=null; //temp's prev's next will be null//
        temp.prev=null; //and then temp's prev will be also null//
    }
}
void InsertBeforeValue(int data, int target)
{
    Node newNode = new Node(data); // Create a new node with the given data

    if (first == null) { // Check if the list is empty
        System.out.println("LinkedList is Empty");
        return;
    }

    if (first.data == target)
    { // Check if the first node has the target value
        newNode.next = first; // Set the new node's next to the first node
        first.prev = newNode; // Set the first node's prev to the new node
        first = newNode; // Update the first node to be the new node
        return;
    }
    Node current = first.next; // Start from the second node
```

DATA STRUCTURES (017013292)

Semester - II

Chapter Name: DOUBLY AND CIRCULAR LINK LIST



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

```
128         while (current != null)
129     {
130         if (current.data == target)
131     {
132             newNode.next = current; // Set the new node's next to the current
133             newNode.prev = current.prev; // Set the new node's prev to the
134             current.node's prev
135             current.prev.next = newNode; // Update the previous node's next to
136             the new node
137             current.prev = newNode; // Update the current node's prev to the new
138             node
139             return;
140         }
141         current = current.next; // Move to the next node
142     }
143     // If the target value is not found
144     System.out.println("Target value not found in the list");
145 }
146 void deleteValue(int value)
147 {
148     if (first == null)
149     {
150         System.out.println("LINKLIST IS EMPTY");
151     }
152     else if (first.data == value)
153     {
154         System.out.println(first.data);
155         first = first.next;
156         first.prev = null;
157     }
158     else
159     {
160         Node temp = first;
161         while (temp.next != null && temp.data != value) {
162             temp = temp.next;
163         }
164         if (temp.data == value)
165         {
166             if (temp.next != null)
167             {
168                 temp.prev.next = temp.next;
169                 temp.next.prev = temp.prev;
170             }
171             else
172             {
173                 temp.prev.next = null;
174                 temp.prev = null;
175             }
176         }
177     }
178 }
179 }
180 class Run11 //created the run class to call the method by making objects//
181 {
182     public static void main(String args[])
183     {
184         DLL d = new DLL();
185         d.displayD();
186         d.insertAtFirst(12);
187         d.displayD();
188         d.insertAtLast(22);
189         d.displayD();
190     }
}
```

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

```
191     d.insertAtFirst(11);
192     d.displayD();
193     d.insertAtLast(222);
194     d.displayD();
195     d.InsertBeforeValue(1,222);
196     d.displayD();
197     d.deleteFirst();
198     d.displayD();
199     d.deleteLast();
200     d.displayD();
201 }
202 }
```