Assignment-03

1) Write a java program to build a Circular Doubly Linked list (Not built-in)?

A+

```
Package Circulardoublylinkedlist;
import java.util.*;
class CircularDoublyLinkedList{
    private class Node{
        int data;
        Node next;
        Node prev;
    public Node(int data){
        this.data =data;
    }
}
Private Node head =null;
public void insertfront(int data){
    Node newNode =new Node(data);
    if(head ==null){
        head = newNode;
        head.next =head;
        head.prev=head;
    }else{
        Node last=head.prev;
        newNode.next=head;
        newNode.prev=last;
        last.next=newNode;
        head.prev = newNode;
        head=newNode;
```

```
public void insertEnd(int data){
    Node newNode = new Node(data);
    if(head == null){
    head = newNode;
    head.next = head;
    head.prev = head;
  }else{
    Node last = head.prev;
    newNode.next = head;
    newNode.prev = last;
    last.next = newNode;
    head.prev = newNode;
  }
}
public void deleteFront(){
    if(head == null){
        System.out.println("List is empty");
        return;
      }
      if(head.next == head){
        head = null;
      }else{
        Node last = head.prev;
        head = head.next;
        head.prev = last;
        last.next = head;
      }
```

```
public void deleteEnd(){
    if(head==null){
        System.out.println("List is empty");
        return;
    }
    if(head.next==head){
        head=null;
    }else{
        Node last=head.prev;
        Node secondLast=last.prev;
        secondLast.next=head;
        head.prev=secondLast;
    }
}

public void display(){
    if(head==null){
        System.out.println("List is empty");
        return;
    }
    Node temp=head;
    do{
        System.out.print(temp.data+" ");
        temp=temp.next;
    }while(temp!=head);
        System.out.println();
    }
```

```
public Void displayReverse(){
    if(head == null){
     System.out.println("List is empty");
     return;
    } Node last=head.prev;
     Node temp= last;
     do{
       System.out.print(temp.data+" ");
       temp=temp.prev;
     } while (temp!= last);
      System.out.println();
    }

public static Void main(String[] args){
 Circular DoublyLinkedList =new CircularDoublyLinkedList
      l.InsertFront(10);
      l.InsertEnd(20);
      l.Insert End(30);
      l.Insert front(5);
   System.out.println("Circular doubly linked list:");
   l.display();
   System.out.println("Circular doubly linked list
                        traversed backward:");
   l.displayReverse();
```

```
System.out.println("After deleting from front:");
l.display();
l.deleteEnd();
System.out.println("After deleting from end:");
l.display();
}
}
```

output:

```
Circular doubly Linked list:
5 10 20 30
Circular doubly linked list traversed backward:
30 20 10 5
After deleting from front:
10 20 30
After deleting from end:
10 20
```

2) Write a Java Program to Merge two Linked List (Both)

A:

```
package mergeLinkedList;

class Node{
    int data;
    Node next;
    public Node(int data){
        this.data = data;
        this.next = null;
    }
}
public class MergeLinkedList{
    public static Node mergeLists(Node head1, Node head2)
        if(head1 == null){
            return head2;
        }
        if(head2 == null){
            return head1;
        }
        Node mergeHead = null;
        Node tail = null;
        while(head1 != null && head2 != null){
            Node newNode;
            if(head1.data < head2.data){
                newNode = head1;
                head1 = head1.next;
```

```
} else {
    newNode = head2;
    head2 = head2.next;
  }
  if (mergeHead == null){
    mergeHead = tail = newNode;
  } else {
    tail.next = newNode;
    tail = newNode;
  }
}
tail.next = (head1 != null)
  return mergeHead;
}
public static void printLast(Node head){
    Node Current = head;
    While (Current != null){
      System.out.print(Current.data + " ");
      Current = Current.next;
    }
    System.out.println();
  }
  public static void main (String [] args){
    Node head1 = new Node(1);
    head1.next.next = new Node(2);
    head1.next.next = new Node(3);
```

```
Node head2 = new Node(4);
head2.next = new Node(5);
head2.next.next = new Node(9);
System.out.print("Linked list 1: ");
printList(head1);
System.out.print("Linked list 2: ");
printList(head2);
Node MergeHead = mergeList(head1, head2);
System.out.println("Merged Linked List: ");
printList(mergeHead);
}
}
```

output

Linked list 1: 1 2 3

Linked list 2: 4 5 9

Merged Linked List: 1 2 3 4 5 9.

3) Write a Java Program to implement Sorted a linked list?

A:

```
package SortedLinkedList{
    class Node{
        int data;
        Node next;
        public Node(int data){
            this.data=data;
            this.next=null;
        }
    }
    Public class SortedLinkedList{
        Node head;
        public void insertSorted(int data){
            Node newNode = new Node(data);
            if(head==null){
                head=newNode;
            }
            if(head.data>=data){
                newNode.next=head;
                head=newNode;
                return;
            }
            Node Current=head;
            While(Current.next!=null && Current.next.
                data(data){
```

```
        Current=Current.next;
    }
    newNode.next=Current.next;
    Current.next=newNode;
}
public void printList(){
    Node Current=head;
    while(Current!=null){
        System.out.print(Current.data+" ");
        Current=Current.next;
    }
    System.out.println();
}
public static void main(String[] args){
    SortedLinkedList l=new SortedLinkedList();
    l.insertSorted(98);
    l.insertSorted(75);
    l.insertSorted(46);
    l.insertSorted(87);
    l.insertSorted(6);
    l.printList();
}
}
```

Output: 6 46 75 87 98