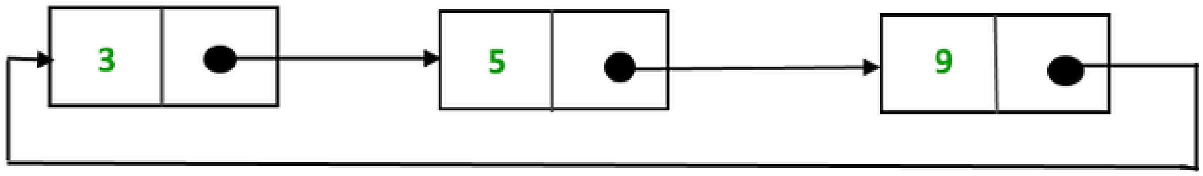
Data Structures and Algorithms
BS (CS/SE)
Lab #03

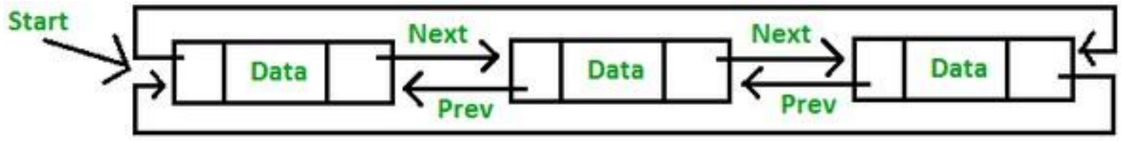
Submission mode: E-Learning **Instructor:** Irum Sindhu

There are generally two types of circular linked lists:

Circular singly linked list: In a circular singly linked list, the last node of the list contains a pointer to the first node of the list. We traverse the circular singly linked list until we reach the same node wherwe started. The circular singly linked list has no beginning or end. No null value is present in the next part of any of the nodes.



Circular Doubly linked list: Circular Doubly Linked List has properties of both doubly linked list and circular linked list in which two consecutive elements are linked or connected by the previous and next pointer and the last node points to the first node by the next pointer and also the first node points to the last node by the previous pointer.



Read more..

```
public class CircularSinglyLinkedListInt {
  static class Node { int data; Node next; Node(int d){ data = d; } }


   Node tail,head;   // tail.next is head when not null
  private int size;    // optional but useful


  public boolean isEmpty(){ return tail == null; }
  public int size(){ return size; }


  // -------- insert at BEGINNING (O(1))
  public void addFirst(int v){
  }


  // -------- insert at END (O(1))
  public void addLast(int v){
  }


  // -------- delete at BEGINNING (O(1))
  public Integer removeFirst(){
  }


  // -------- delete at END (O(n))
  public Integer removeLast(){


  }
```

Exercise

1. Understand provided code and implement all required methods (with all possible exceptions) in DoubleLinkedList.

```
public class Node { String
        name; Node prev, next;


        Node (String name)
        {
                this.prev = null; this.next =
                null; this.name = name;
        }
}

public class DoubleLinkedList { Node head;

        // Add node with name in beginning of linkedlist, name as parameter
        public void insertAtBeginning(String name)
        {


        }
        // Add node in beginning of linkedlist, node as parameter public void insertAtBeginning(Node
        node)
        {

        }
        // Add node in end of linkedlist, name as parameter public void insertAtEnd(String
        name)
        {


        }
        // Add node in end of linkedlist, node as parameter public void insertAtEnd(Node
{


}
// Add node before name which is provided as param , name and node as params
public void insertBeforeName(String name, Node node)
{

}


// Make double linkedlist as Circular Double LinkedList public void makeCircular()
{


}


// Print all the nodes in linkedlist, make sure it works on circular double linkedlist
public void printAll()
{


}
// Test the class
```

2. In previous labs, you have designed single linkedlist with all possible common methods with only head.

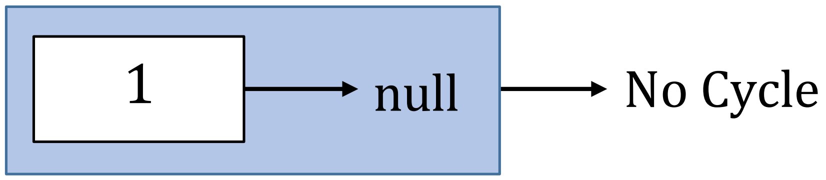
Now your task is to implement following methods but this time you have to make another variable say **tail** for accessing last element directly.

- All types of methods for inserting (Beginning, End)
- All types of methods for removing (Beginning, End)

Compare these methods with those which were designed without **tail**.

3. Implement both Circular linked list (with or without tail) and its all operation.

4. Design a method that takes head as param and detect whether linked list contains cycle or not? Cycle exists in a linked list if any node is visited twice while traversing whole traversing.


1
null
No Cycle