Merit-Quality-Excellence

Sukkur IBA University Khairpur Campus

Data Structures

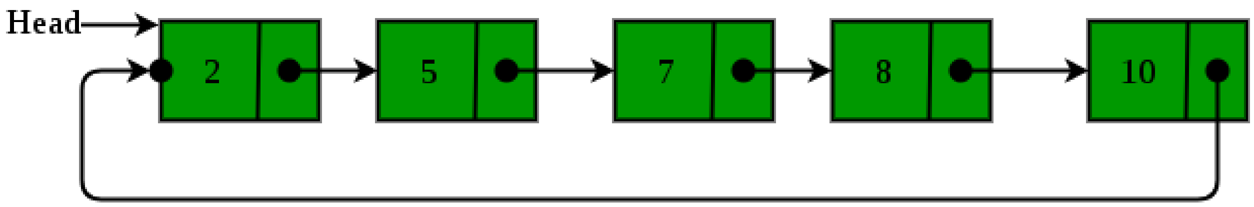
LAB No: 03

Instructor: Marina Gul

Objective of Lab No. 3:
After performing lab3, students will be able to:
- doubly Linked list
- Circular Linked List

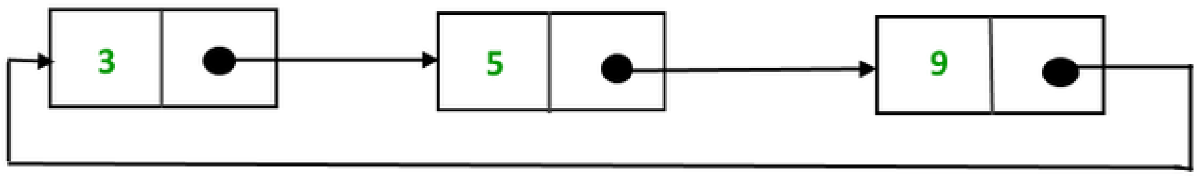
Circular linked list

The circular linked list is a linked list where all nodes are connected to form a circle. In a circular linked list, the first node and the last node are connected to each other which forms a circle. There is no NULL at the end.

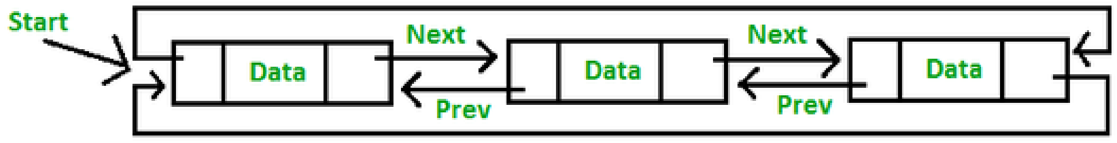


There are generally two types of circular linked lists:

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 where we 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..

```
class Node{
    int data;
    Node next;
    Node prev;

    //Constructor
    Node(int data)
    {
        this.data=data;
        this.next=null;

    }
class CircularLinkedList {
    Node head;

    void addLast(int data) {
        Node node = new Node(data);

        if (head == null)
            head = node;

        node.next = head;

        else{
            Node n = head;
            while (n.next != head)
                n = n.next;

            n.next = node;
        }
    }

}
```

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 node)
    {

    }
    // Add node after name which is provided as param , name and
node as params
    public void insertAfterName(String name, Node 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
    public static void main(String[] args) {
        // Test all above methods

    }

}
```

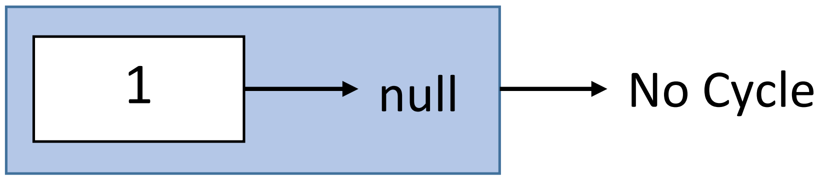
2. In previous labs, you have designed single linkedlist with all possible common methods with only head.

Now your task is to implement following methods (Single/Double LL) 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. 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