Aror University of Art, Architecture, Design and Heritage
SUKKUR,Sindh
Department of Multimedia and Gaming
Course: Data Structures CSC-221 (Practical)
Instructor: Engr. Fatima Jaffar

LAB 06

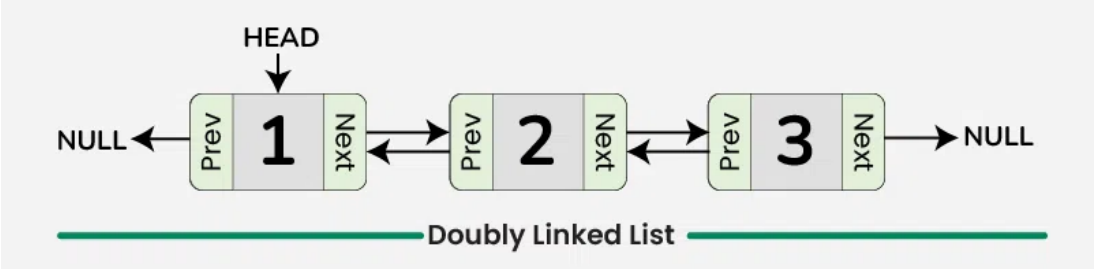
Objective: Understanding and implementing doubly linked list and Circular Linked List in java

Introduction:

- A doubly linked list is a more complex data structure than a singly linked list, but it offers several advantages.
- The main advantage of a doubly linked list is that it allows for efficient traversal of the list in both directions.
- This is because each node in the list contains a pointer to the previous node and a pointer to the next node.
- This allows for quick and easy insertion and deletion of nodes from the list, as well as efficient traversal of the list in both directions.

What is a Doubly Linked List?

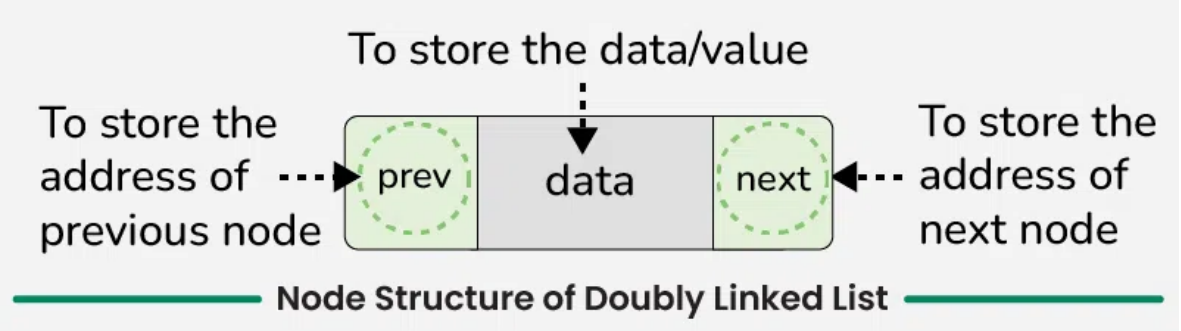
A doubly linked list is a data structure that consists of a set of nodes, each of which contains a value and two pointers, one pointing to the previous node in the list and one pointing to the next node in the list. This allows for efficient traversal of the list in both directions, making it suitable for applications where frequent insertions and deletions are required**.**



Representation of Doubly Linked List in Data Structure

In a data structure, a doubly linked list is represented using nodes that have three fields:

1. Data
2. A pointer to the next node (next)
3. A pointer to the previous node (prev)



Node Definition

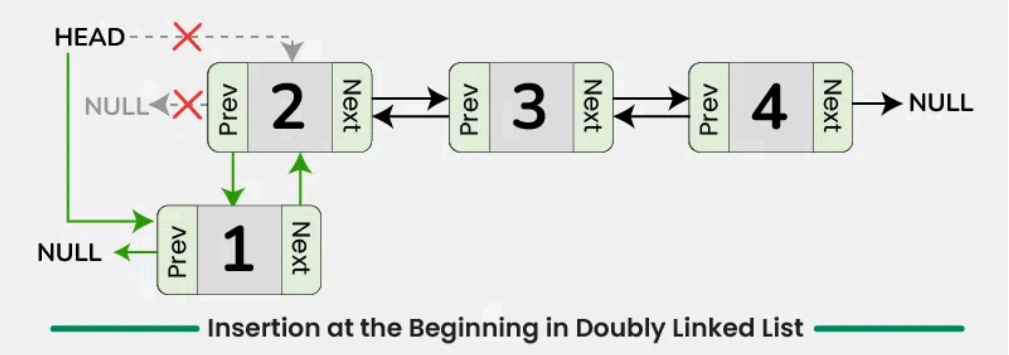
Here is how a node in a Doubly Linked List is typically represented:

```
class Node {

    int data;
    Node prev;
    Node next;

    // Constructor
    Node(int d) {
        data = d;
        prev = next = null;
    }
}
```

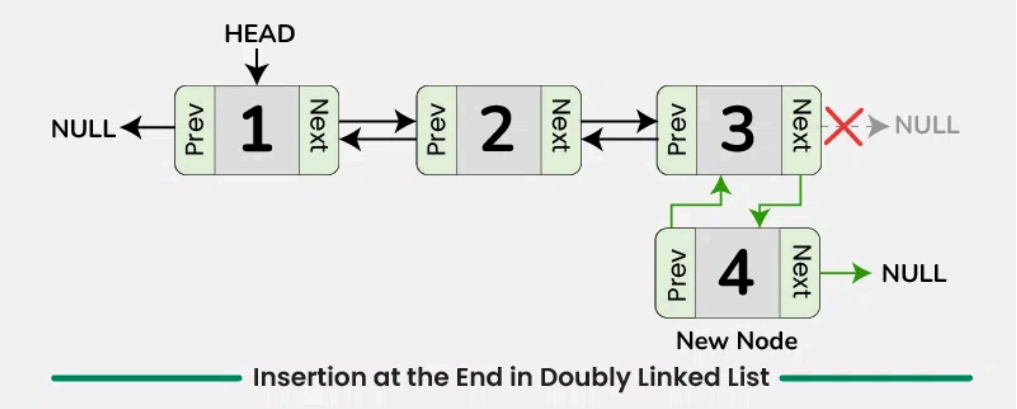
Operations on Doubly Linked List

- Traversal in Doubly Linked List
- Searching in Doubly Linked List
- Finding Length of Doubly Linked List

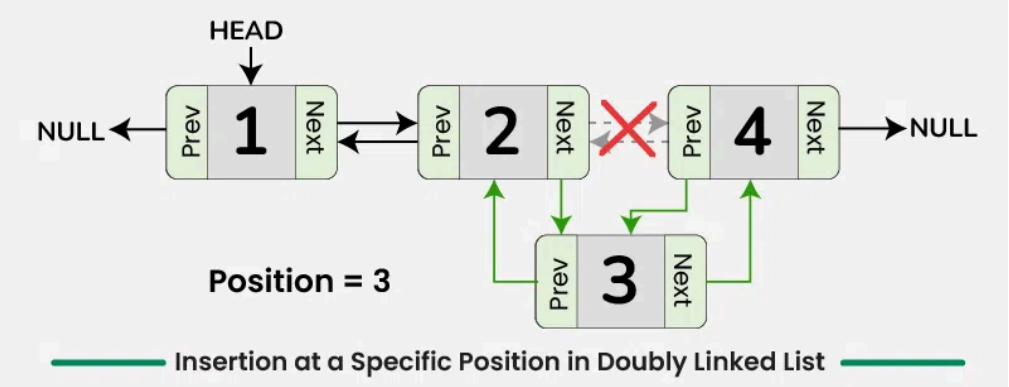
- **Insertion in Doubly Linked List:**
  - Insertion at the beginning of Doubly Linked List



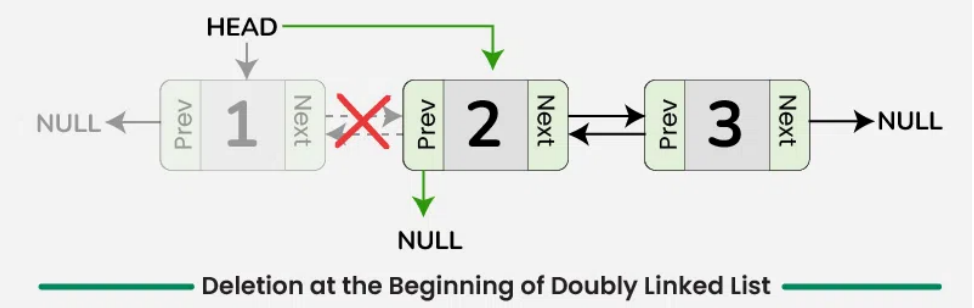
  - Insertion at the end of the Doubly Linked List



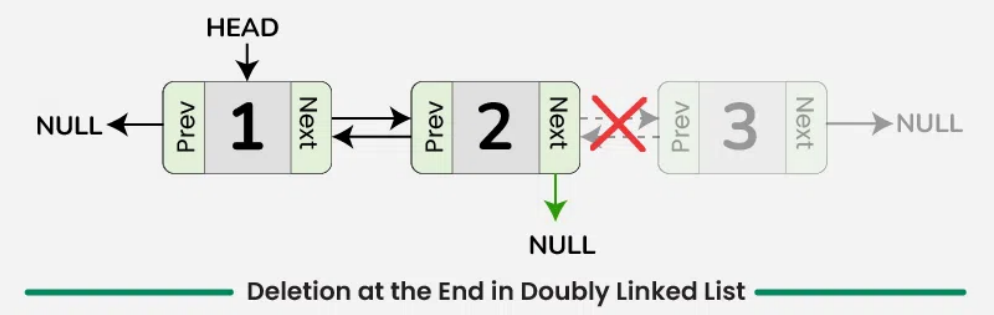
  - Insertion at a specific position in Doubly Linked List



- ## Deletion in Doubly Linked List:
    - **Deletion of a node at the beginning of Doubly Linked List**



    - **Deletion of a node at the end of Doubly Linked List**



    - **Deletion of a node at a specific position in Doubly Linked List**

Traversal in Doubly Linked List

To Traverse the doubly list, we can use the following steps:

a. Forward Traversal:

- Initialize a pointer to the head of the linked list.
- While the pointer is not null:
    - Visit the data at the current node.
    - Move the pointer to the next node.

b. Backward Traversal:

- Initialize a pointer to the tail of the linked list.
- While the pointer is not null:
    - Visit the data at the current node.
    - Move the pointer to the previous node.

LAB TASKS

1. Write java code of all the operations mentioned in the manual

   - **Create a doubly linked list:**
     Implement a function to create an empty doubly linked list.
   - **Insert a node:**
     Implement functions to insert a node at the beginning, end, or a specific position in the list.
   - **Delete a node:**
     Implement functions to delete a node at the beginning, end, or a specific position in the list.
   - **Search for a node:**
     Implement a function to search for a node with a given value.
   - **Traverse the list:**

     Implement functions to traverse the list in both forward and backward directions.

2. Write a java program to find the length of a doubly linked list
3. Implement circular linked list using java
4. Insert first and last node in circular linked list
5. Insert last node in circular linked list