Linked List 3

[Intro to Doubly Linked List](#_302cpkj0ui7a)

[Advantages of Doubly Linked List](#_laeoduu4dbdu)

[Disadvantages of Doubly Linked List](#_88zb9kwqnoq7)

[Applications of Doubly Linked List](#_8efhwx86jzxs)

[Insertion](#_z6o8hqsbtttj)

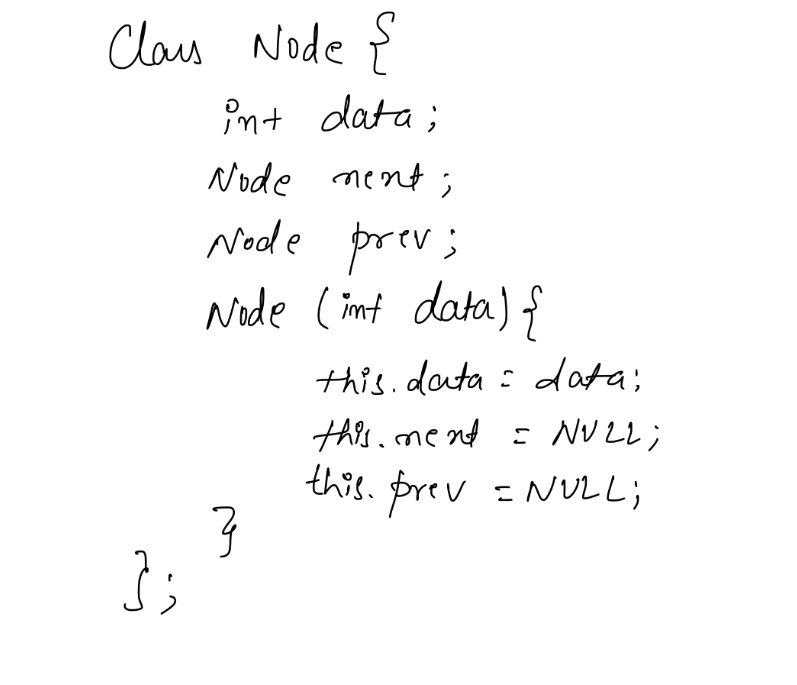
[Deletion](#_2b58ktjeor6f)

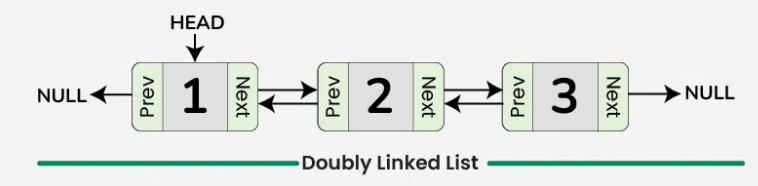
[LRU Cache](#_jn30q51ovpyo)

[Clone a doubly Linked List](#_wpaglf2xd9i7)

# 

# Intro to Doubly Linked List

* A **Doubly Linked List** **(DLL)** is a two-way list in which each node has two pointers, the next and previous that have reference to both the next node and previous node respectively. Unlike a singly linked list where each node only points to the next node, a doubly linked list has an extra previous pointer that allows traversal in both the forward and backward directions.



## 

## Advantages of Doubly Linked List

* **Efficient traversal in both directions:** Doubly linked lists allow for efficient traversal of the list in both directions, making it suitable for applications where frequent insertions and deletions are required.
* **Easy insertion and deletion of nodes:** The presence of pointers to both the previous and next nodes makes it easy to insert or delete nodes from the list, without having to traverse the entire list.
* **Can be used to implement a stack or queue:** Doubly linked lists can be used to implement both stacks and queues, which are common data structures used in programming.

## Disadvantages of Doubly Linked List

* **More complex than singly linked lists:**
* **More memory overhead:**

## Applications of Doubly Linked List

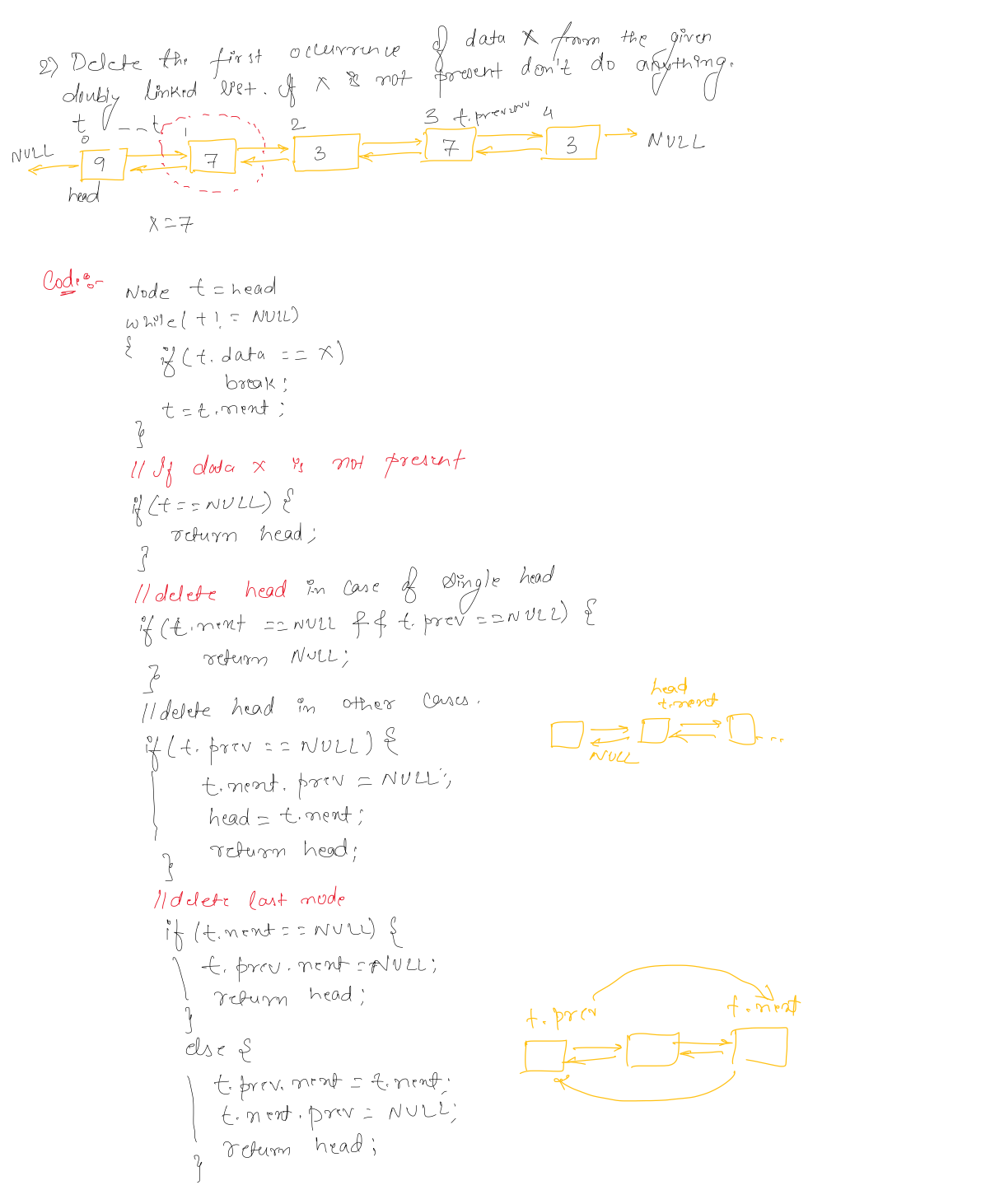
* Implementation of undo and redo functionality in text editors.
* Cache implementation where quick insertion and deletion of elements are required.
* Browser history management to navigate back and forth between visited pages.
* Music player applications to manage playlists and navigate through songs efficiently.
* Implementing data structures like [Deque](https://www.geeksforgeeks.org/deque-set-1-introduction-applications) (double-ended queue) for efficient insertion and deletion at both ends.

# Insertion

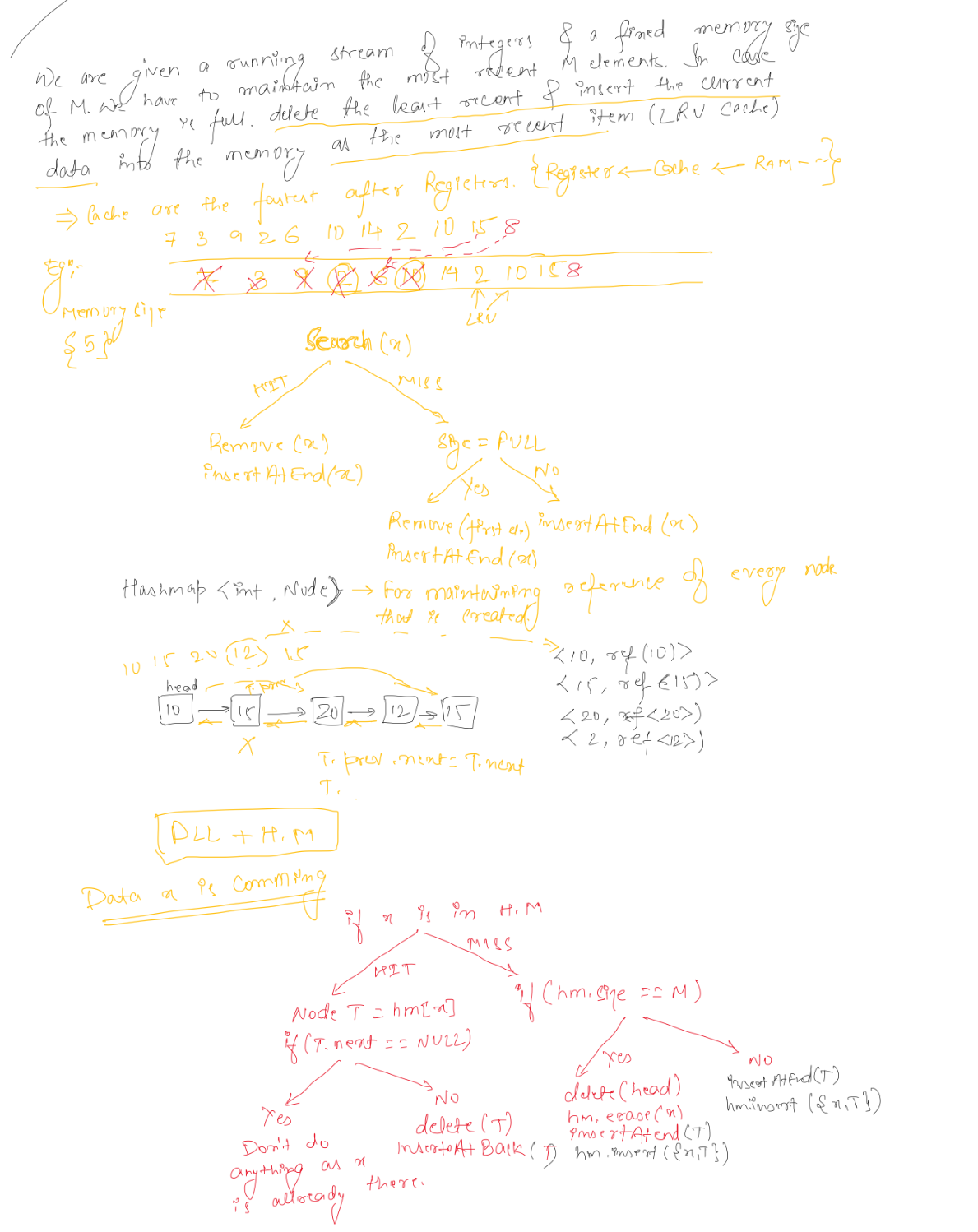
# 

# 

# Deletion



# LRU Cache



# Clone a doubly Linked List

