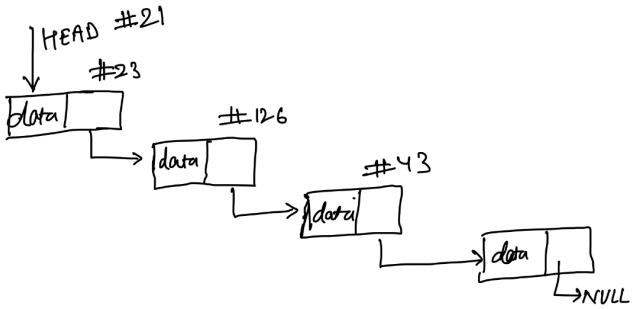
LinkedList1

Agenda

Intro To Linked List
Access Kth Element
Find An Element
Insert a new Node With data v at index p
Delete the first occurrence of a given element
Reverse the linked list
Check Palindrome

Intro To Linked List

- A **linked list** is a linear data structure that consists of a series of nodes connected by pointers (in C or C++) or references (in Java, Python and JavaScript).
- Each node contains **data** and a **pointer/reference** to the next node in the list.
- Unlike **arrays, linked lists** allow for efficient **insertion** or **removal** of elements from any position in the list, as the nodes are not stored contiguously in memory.

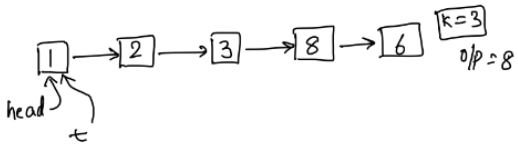



```
Class Node {
    int data;
    Node nt;
    Node (int n) {
        data = n;
        next = NULL;
    }
};
```

- A node class consists of a data parameter and reference/pointer for the next memory location.
- Along with the constructor which initializes each node.

Access Kth Element

Given a linked list, find the element at location **k.**

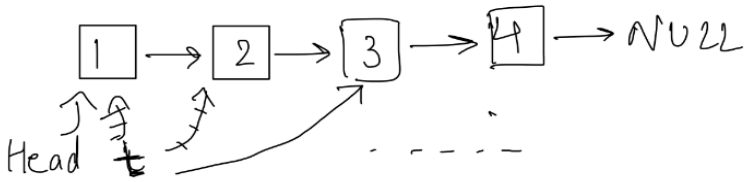



```
Node t = head;
//Do iteration for k times
while (k >0){
   t=t.next; //keep moving to next node
   k--;
}
if (t != NULL)    // k=5 can be null, but our pointer will go till there
    return t.data;
else
    return -1;
```

```
//Whenever you try to access t.next /t.null make sure that
 t != NULL, else we will get NPE.
```

Find An Element

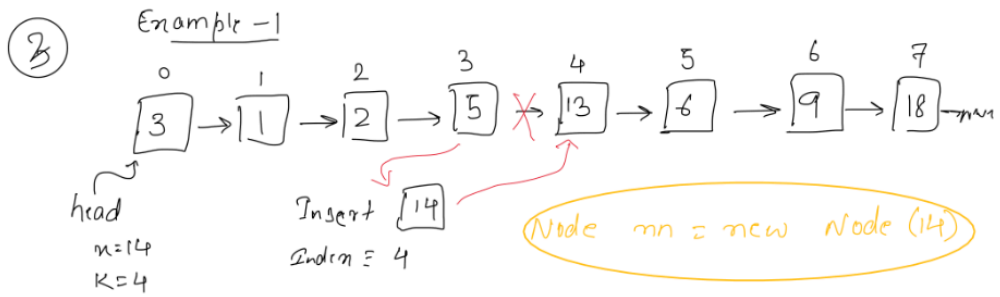
Given a linked list, find a given element **k.**




```
Node t = Head;
while (t != NULL) {  // Iterate until t becomes NULL
    if (t.data == k) {  // If data is found
        return true;
    }
    t = t.next;  // move reference to next.
}
```

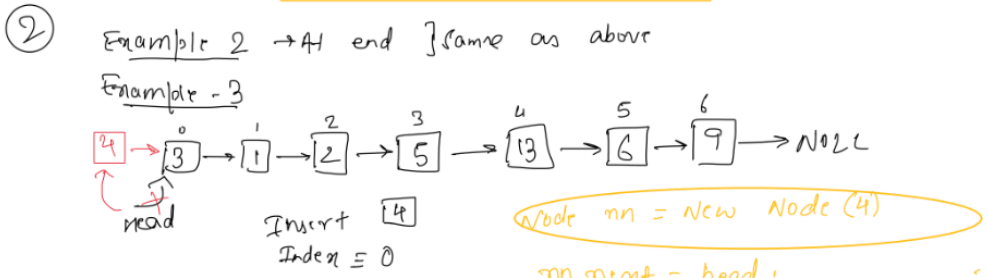
Insert a new Node With data v at index p

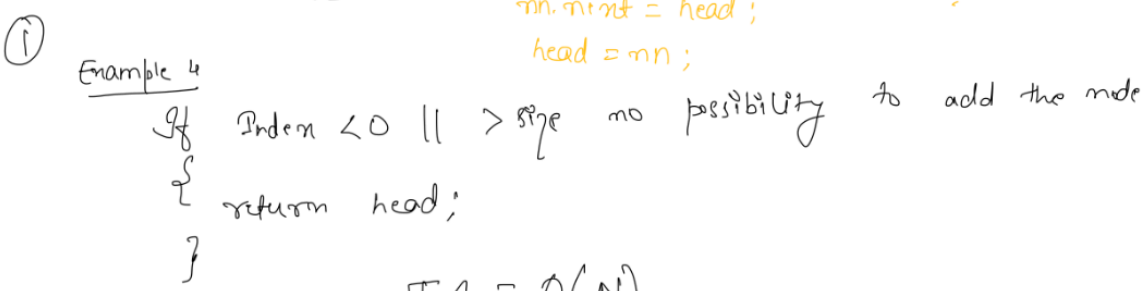
Given a linked list insert a new node with value v at index p



1) Jump K-1 times
2) Create Node & assign

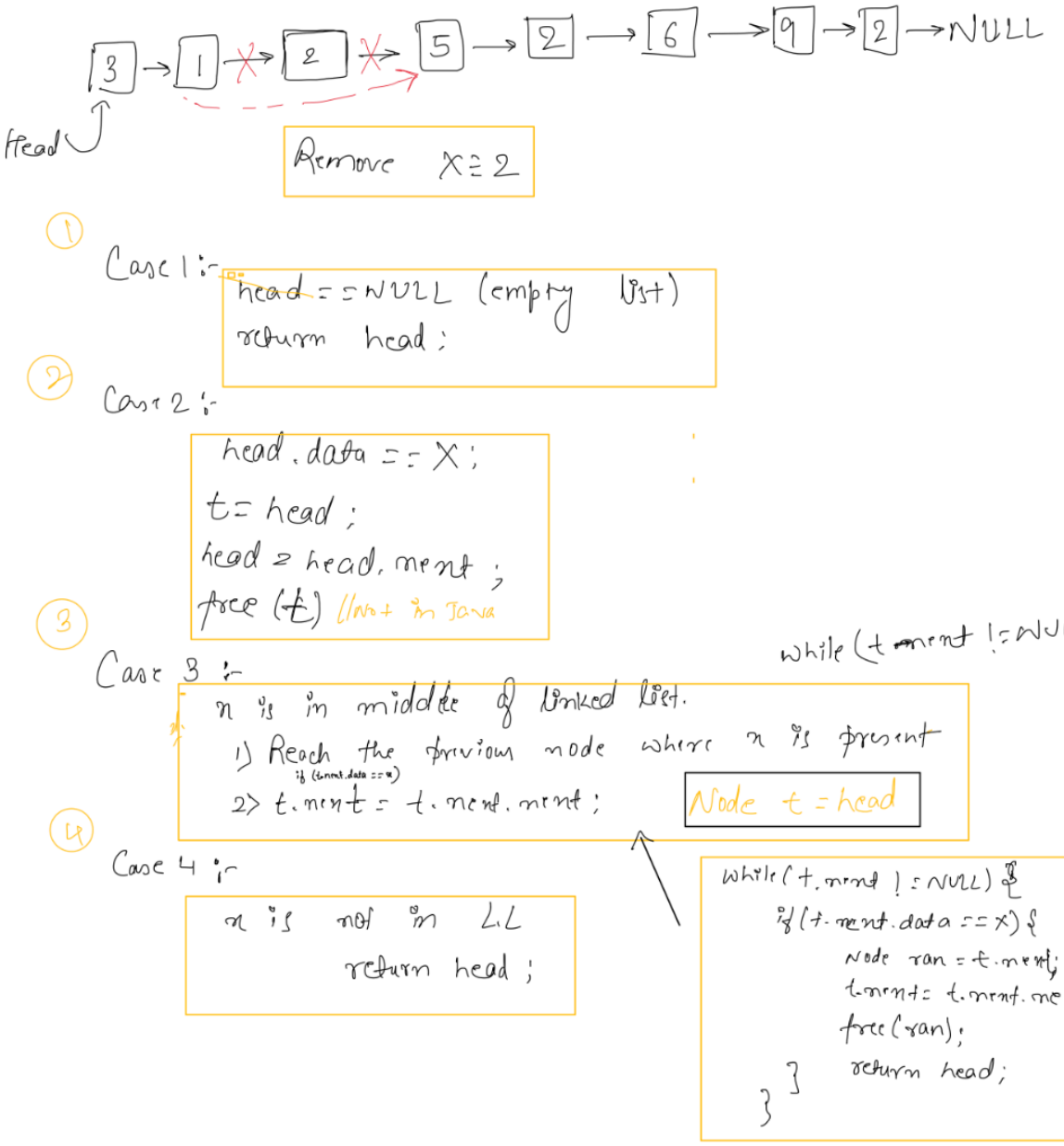
```
Node nn = new Node (14);
nn.next = t.next;
t.next = nn
```





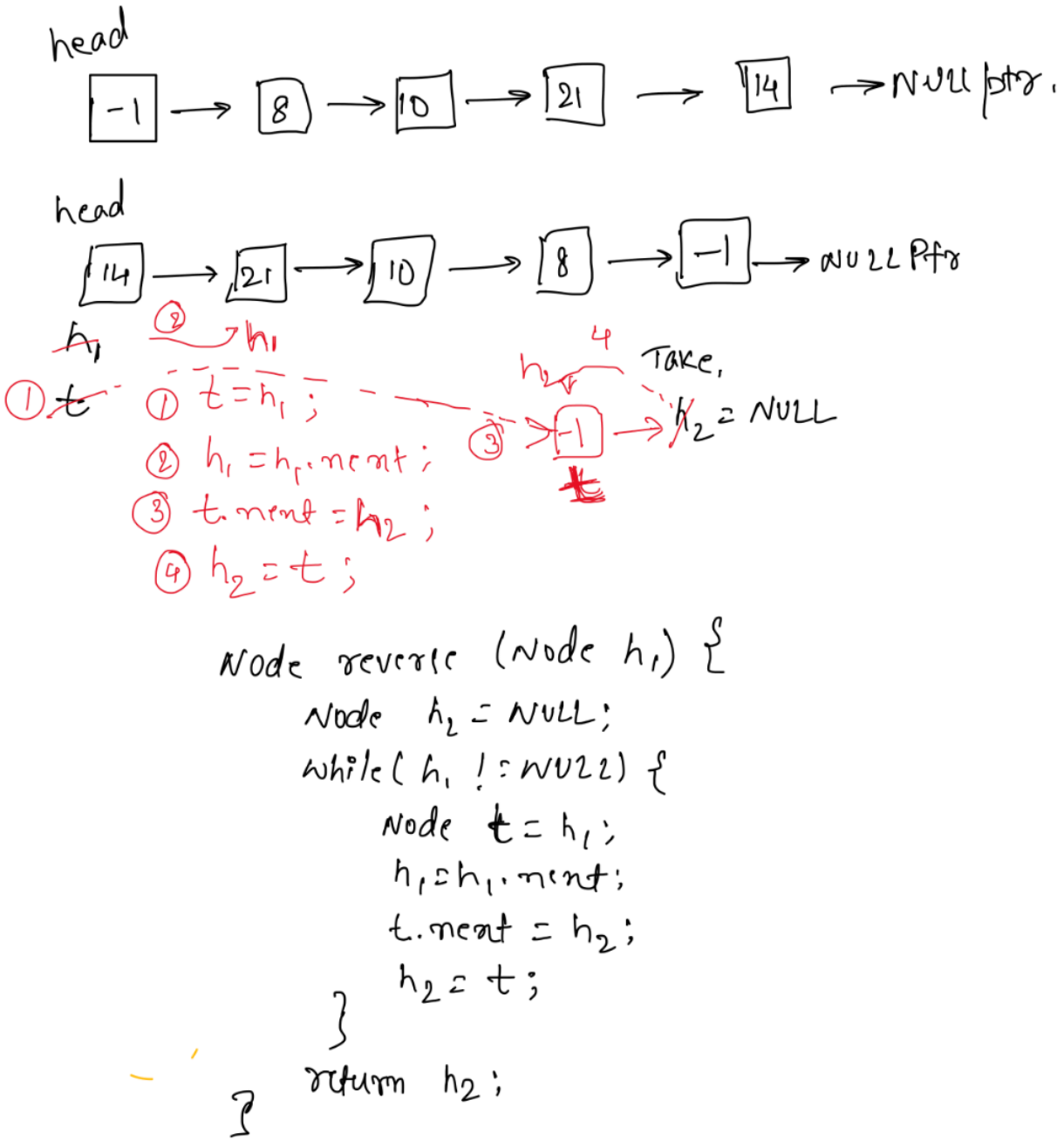
Delete the first occurrence of a given element

Delete the first occurrence of value X in a linked list, if ele is not present, leave it as it is.



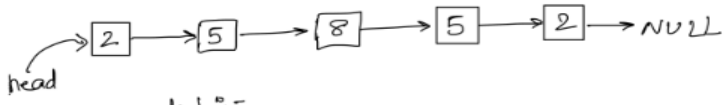
Reverse the linked list

Reverse the given linked list



Check Palindrome

Check whether the given linked list forms the palindrome



Approach 1 :-

① Create a copy of the given LL & reverse it

② Compare both the linked list

T.C = O(N)

S.C = O(N)

Approach 2 :-

Middle

Reverse 1st half & compare both part.

① Find the middle element

N = even (10)

1-5 | 6 — 10

reverse (ref of LL at 6)

N = odd (9)

1-5-4-3-6-3-4-5-1

1-5-4-3

1-5-4-3-6

```
N = size(N)
Node t = head;
for(int i=0; i<(N+1)/2; i++){
    t = t.next;
}
h2 = reverse(t);
t1 = h1; t2 = h2;
for(int i=0; i<N/2; i++){
    if(t1.data != t2.data)
        return false
    t1 = t1.next;
    t2 = t2.next;
}return true;
```

Applications Of Linked List

Applications of linked list in computer science:

1. Implementation of stacks and queues
2. Implementation of graphs: Adjacency list representation of graphs is the most popular which uses a linked list to store adjacent vertices.
3. Dynamic memory allocation: We use a linked list of free blocks.
4. Maintaining a directory of names
5. Performing arithmetic operations on long integers
6. Manipulation of polynomials by storing constants in the node of the linked list
7. Representing sparse matrices

Applications of linked list in the real world:

1. Image viewer – Previous and next images are linked and can be accessed by the next and previous buttons.
2. Previous and next page in a web browser – We can access the previous and next URL searched in a web browser by pressing the back and next buttons since they are linked as a linked list.
3. Music Player – Songs in the music player are linked to the previous and next songs. So you can play songs either from starting or ending of the list.
4. GPS navigation systems- Linked lists can be used to store and manage a list of locations and routes, allowing users to easily navigate to their desired destination.
5. Robotics- Linked lists can be used to implement control systems for robots, allowing them to navigate and interact with their environment.
6. Task Scheduling- Operating systems use linked lists to manage task scheduling, where each process waiting to be executed is represented as a node in the list.
7. Image Processing- Linked lists can be used to represent images, where each pixel is represented as a node in the list.
8. File Systems- File systems use linked lists to represent the hierarchical structure of directories, where each directory or file is represented as a node in the list.
9. Symbol Table- Compilers use linked lists to build a symbol table, which is a data structure that stores information about identifiers used in a program.

10. Undo/Redo Functionality- Many software applications implement undo/redo functionality using linked lists, where each action that can be undone is represented as a node in a doubly linked list.
11. Speech Recognition- Speech recognition software uses linked lists to represent the possible phonetic pronunciations of a word, where each possible pronunciation is represented as a node in the list.
12. Polynomial Representation- Polynomials can be represented using linked lists, where each term in the polynomial is represented as a node in the list.
13. Simulation of Physical Systems- Linked lists can be used to simulate physical systems, where each element in the list represents a discrete point in time and the state of the system at that time.

Applications of Circular Linked Lists:

1. Useful for implementation of a queue. Unlike this implementation, we don't need to maintain two-pointers for the front and rear if we use a circular linked list. We can maintain a pointer to the last inserted node and the front can always be obtained as next of last.
2. Circular lists are useful in applications to go around the list repeatedly. For example, when multiple applications are running on a PC, it is common for the operating system to put the running applications on a list and then cycle through them, giving each of them a slice of time to execute, and then making them wait while the CPU is given to another application. It is convenient for the operating system to use a circular list so that when it reaches the end of the list it can cycle around to the front of the list.
3. Circular Doubly Linked Lists are used for the implementation of advanced data structures like the Fibonacci Heap.
4. Circular linked lists can be used to implement circular queues, which are often used in operating systems for scheduling processes and managing memory allocation.
5. Used in database systems to implement linked data structures, such as B+ trees, which are used to optimize the storage and retrieval of data.
6. Circular linked lists can be used in networking. For instance, to implement circular buffers for streaming data, such as video and audio, in networking applications.

7. Video games use circular linked lists to manage sprite animations. Each frame of the animation is represented as a node in the list, and the last frame is connected to the first frame to create a loop.
8. Circular linked lists can be used to represent a buffer of audio or signal data in signal processing applications. The last node is connected to the first node to create a loop, and the processing algorithms can efficiently iterate over the data.
9. Traffic light control systems use circular linked lists to manage the traffic light cycles. Each phase of the traffic light cycle is represented as a node in the list, and the last node is connected to the first node to create a loop.

Application of Doubly Linked Lists:

1. Redo and undo functionality.
2. Use of the Back and forward button in a browser.
3. The most recently used section is represented by the Doubly Linked list.
4. Other Data structures like Stack, Hash Table, and Binary Tree can also be applied by Doubly Linked List.
5. Used to implement game objects and their interactions in a game engine.
6. Used in networking.
7. Used in Graph algorithms.
8. Operating systems use doubly linked lists to manage the process scheduling. Each process waiting to be executed is represented as a node in the doubly linked list, and the operating system can easily traverse the list in both directions to manage the process queue.