Day 22 (07/07/2025)

---

1. Middle of a Linked List

A problem that introduces linked list traversal techniques and teaches how to find the middle element efficiently using the two-pointer approach.

Given the **head of a linked list**, the task is to find the **middle node**. For example, the middle of 1->2->3->4->5 is 3. If there are **two middle nodes** (even count), return the **second middle**. For example, middle of 1->2->3->4->5->6 is 4. This problem appears frequently in **interviews** and **real-world applications** like implementing efficient search algorithms or dividing linked lists for processing. You can solve this by first counting nodes and then traversing again, but try to think of **more efficient approaches** using the two-pointer technique.

This teaches **two-pointer traversal** and **optimal linked list navigation** techniques that are essential for **efficient list processing and divide-and-conquer algorithms**.

Your task: Find the middle node of a linked list using efficient traversal techniques without multiple passes.

Examples

Input:

Linked list: 1->2->3->4->5

Output:

3

Input:

Linked list: 2->4->6->7->5->1

Output:

7

---

2. Frequency in a Linked List

A problem that demonstrates linked list traversal and counting techniques, teaching how to search and count occurrences of specific elements.

Given a **singly linked list** and a **key**, count the number of occurrences of the given key in the linked list. This operation is fundamental in **data analysis** and **frequency counting** applications where you need to **determine how often specific values appear** in sequential data structures. The challenge involves understanding how to traverse the entire list while maintaining an accurate count of matches.

This introduces **sequential search algorithms** and **frequency analysis techniques** that are crucial for **data processing and statistical analysis of linked data structures**.

Your task: Count occurrences of a specific key in a linked list using efficient traversal and counting methods.

Examples

Input:

Linked List: 1->2->1->2->1->3->1, key = 1

Output:

4

Input:

Linked List: 1->2->1->2->1, key = 3

Output:

0

---

3. Print Linked List

A problem that teaches basic linked list traversal and output formatting, demonstrating fundamental operations for displaying linked list contents.

Given a **linked list**, print all the elements of the linked list **separated by space**. This is a fundamental operation in **linked list manipulation** and serves as the building block for more complex linked list algorithms. Understanding how to properly traverse and display linked list contents is essential for **debugging** and **data visualization** in real-world applications.

This teaches **basic linked list traversal** and **output formatting techniques** that are essential for **linked list debugging and data presentation**.

Your task: Traverse and print all elements of a linked list in the correct order with proper spacing.

Examples

Input:

LinkedList: 1 -> 2

Output:

1 2

Input:

Linked List: 49 -> 10 -> 30

Output:

49 10 30