Deletion in Linked List - GeeksforGeeks

Source: https://www.geeksforgeeks.org/deletion-in-linked-list/

Courses Tutorials Practice Jobs DSA Tutorial Interview Questions Quizzes Must Do Advanced DSA System Design Aptitude Puzzles Interview Corner DSA Python Technical Scripter 2026 Explore DSA Fundamentals Logic Building Problems Analysis of Algorithms Data Structures Array Data Structure String in Data Structure Hashing in Data Structure Linked List Data Structure Stack Data Structure Queue Data Structure Tree Data Structure Graph Data Structure Trie Data Structure Algorithms Searching Algorithms Sorting Algorithms Introduction to Recursion Greedy Algorithms Tutorial Graph Algorithms Dynamic Programming or DP Bitwise Algorithms Advanced Segment Tree Binary Indexed Tree or Fenwick Tree Square Root (Sqrt) Decomposition Algorithm Binary Lifting Geometry Interview Preparation Interview Corner GfG160 Practice Problem GeeksforGeeks Practice - Leading Online Coding Platform Problem of The Day - Develop the Habit of Coding DSA Course 90% Refund Deletion in Linked List Last Updated : 19 Sep, 2025 Deleting a node in a Linked List is an important operation and can be done in three main ways: removing the first node, removing a node in the middle, or removing the last node. In this article, we will explore deletion operation on Linked List for all the above scenarios. 1. Deletion at the Beginning of Linked List Deletion at the Beginning operation involves removing the first node of the linked list. Check if the list is empty : If the head is NULL , the list is empty, and there's nothing to delete. Update the head pointer : Set the head to the second node (head = head->next). Delete the original head node : The original head node is now unreferenced, and it can be freed/deleted if necessary (in languages like C++ where memory management is manual). To read more about Deletion at the Beginning Refer, Deletion at beginning (Removal of first node) in a Linked List Try it on GfG Practice 2. Deletion at Specific Position of Linked List Deletion at a specified position in a linked list involves removing a node from a specific index/position, which can be the first, middle, or last node. Check if the position is valid : If the position is out of bounds (greater than the length of the list), return an error or handle appropriately. Traverse the list to find the node just before the one to be deleted: Start from the head and move through the list until reaching the node at position n-1 (one before the target position). Update the next pointer : Set the next pointer of the (n-1)■■ node to point to the node after the target node (node_to_delete->next). Delete the target node : The node to be deleted is now unreferenced, and in languages like C++ or Java, it can be safely deallocated. To read more about Deletion at specific position Refer, Delete a Linked List node at a given position 3. Deletion at the End of Linked List Deletion at the end operation involves removing the last node of the linked list. Check if the list is empty : If the head is NULL , the list is empty, and there's nothing to delete. If the list has only one node : Simply set the head to NULL (the list becomes empty). Traverse the list to find the second-last node: Start from the head and iterate through the list until you reach the second-last node (where the next of the node is the last node). Update the next pointer of the second-last node: Set the second-last node’s next to NULL (removing the link to the last node). Delete the last node : The last node is now unreferenced and can be deleted or freed, depending on the language used. To read more about Deletion at the end Refer, Deletion at end (Removal of last node) in a Linked List Comment Article Tags: Article Tags: Linked List DSA