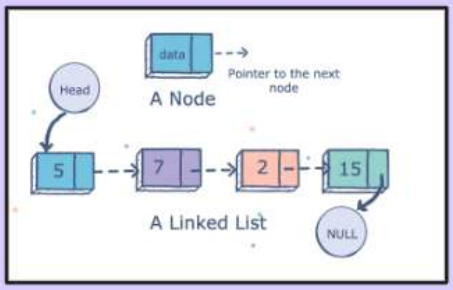
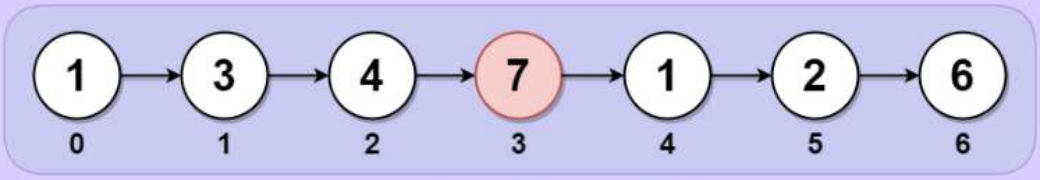
Delete the Middle Node of a Linked List (Leetcode-2095)

@manojofficialmj





amazon Microsoft Meta

2095 Leetcode

Delete the middle Node of Linked List

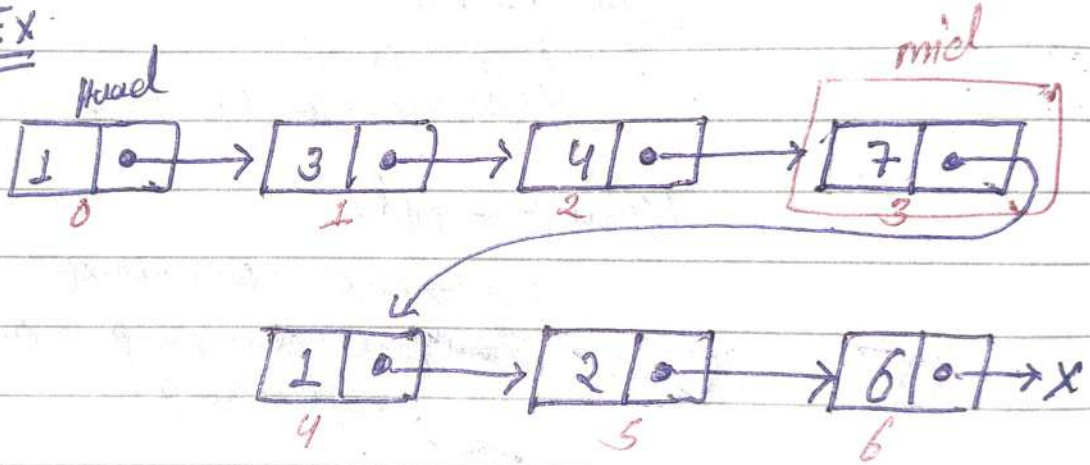
Important line

Size of LL (N)	middle Node
1	0
2	1
3	1
4	2
5	2

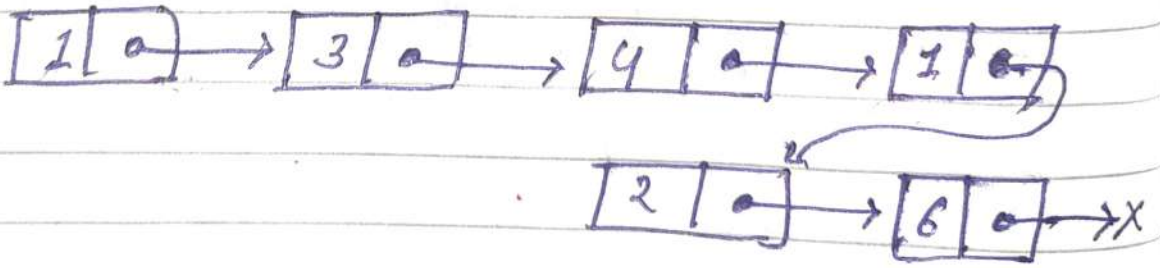
middle Node = $(N/2)^{th}$

Ex

Input



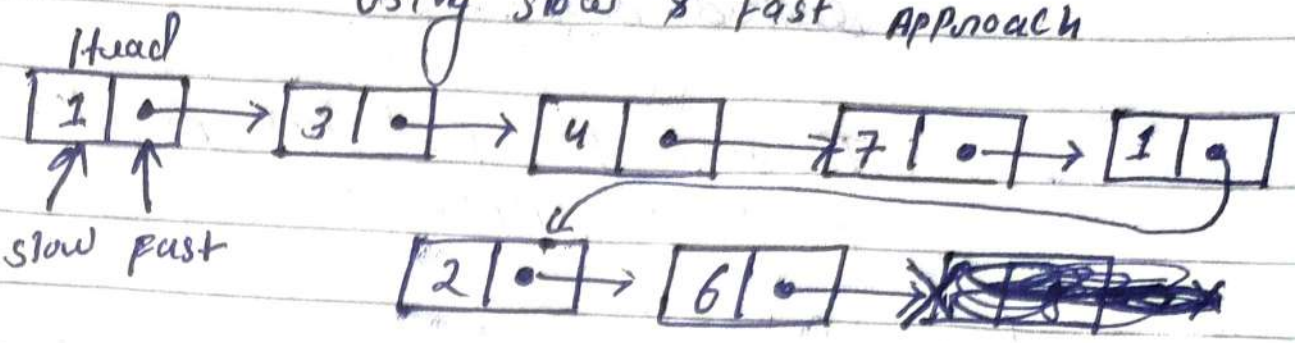
Output



N = 7 { Mid = N/2

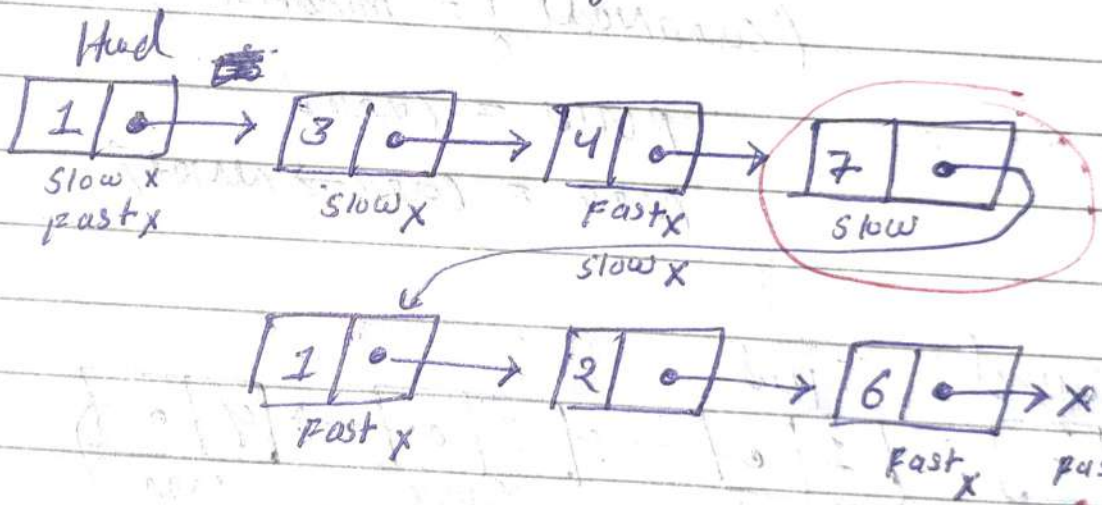
STEP 1

Approach 1 First Find the middle node using slow & Fast Approach



slow 1 step chalega to fast 2 step chalega

→ Note jab fast 2 step chal lega tabhi slow 1 step aage chalega



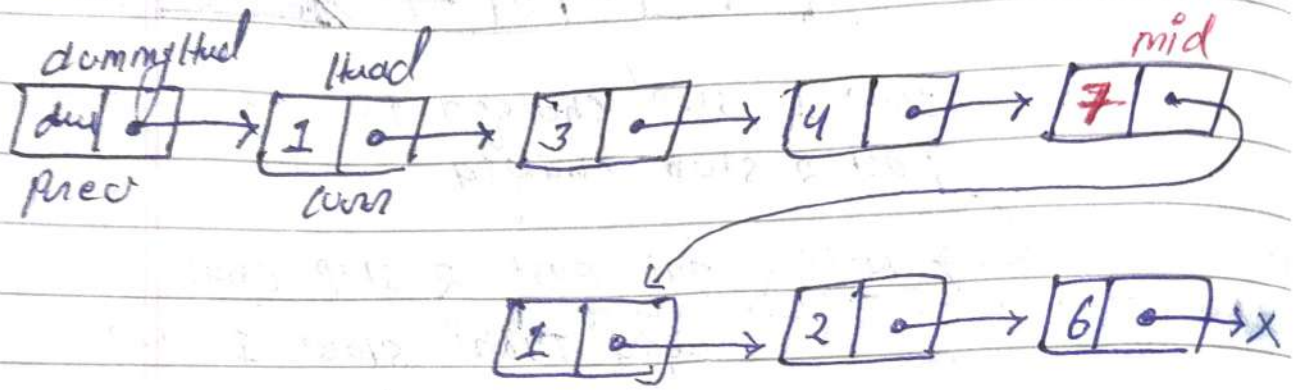
(isko mai ne name (middle node) slow node find kr chuka hai)

fast 1 step hi move kr rahi hai

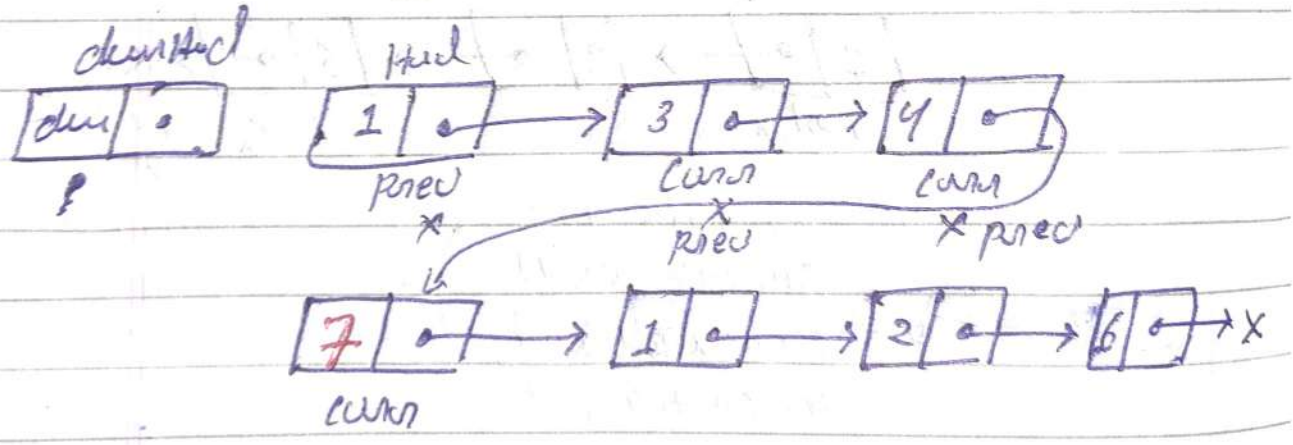
slow = middleNode

STEP2 Delete the middle Node from linked list

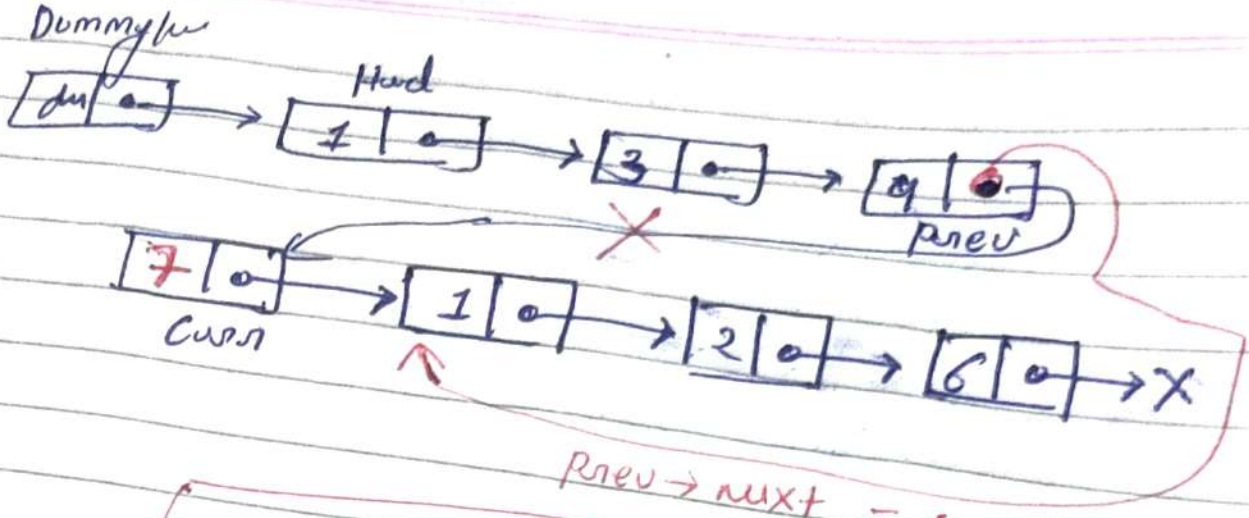
Traversal to find the middle Node & delete it.




```
(currNode != middNode) {
    prev = curr;
    curr = curr -> next;
}
```




```
(curr == middNode) {
    prevNode -> next
    delete curr;
    break
}
```



Prev → next = curr → next

return dummyHead → next;

Output

Time complexity = O(N)

Space complexity = O(1)

```cpp
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* middleNode(ListNode* head) {
        ListNode* slow = head;
        ListNode* fast = head;

        while(fast != NULL){
            fast = fast->next;
            if(fast != NULL){
                fast = fast->next;
                // Ab fast two step chal chuka hai
                // Ab hum slow ko one step chla skte hai
                slow = slow->next;
            }
        }
        return slow;
    }

    ListNode* deleteMiddle(ListNode* head) {
        if(head->next == NULL){
            return NULL;
        }

        // Step 1: Find the middle node from linked list
        ListNode* midNode = middleNode(head);

        // Step 2: Delete the middle node from linked list
        ListNode* dummyHead = new ListNode();
        dummyHead->next = head;
        ListNode* prevNode = dummyHead;
        ListNode* currNode = head;

        while(currNode != NULL){
            if(currNode != midNode){
                prevNode = currNode;
                currNode = currNode->next;
            }
            else if(currNode == midNode){
                prevNode->next = currNode->next;
                delete currNode;
                break;
            }
        }

        head = dummyHead->next;
        delete dummyHead;
        return head;
    }
};
```