Algorithm 1. *Find N^{th} node from the end of a Singly Linked List without using the number of elements of linked list (not allowed to store the count also). Time: $O(n)$ and space: $O(1)$.*

Algorithm 2. *Find the middle element of the non-empty singly linked list. Middle element for list of even length n is the element at $\frac{n}{2}^{th}$ index from start of list and for odd n, element is present at $\frac{n+1}{2}^{th}$ index. Find this element without using or storing the variable n. Time: $O(n)$ and space: $O(1)$.*

Algorithm 3. *Detect whether a loop is present in a singly linked list, given only head of that list. Obviously tail and counts are not given. Loop in a linked list can be formed when next pointer of tail refers to some element already present in the list. (Floyd's Cycle-Finding Algorithm)*

Algorithm 4. *Find intersection point of two linked lists. Time: $O(m+n)$ and space: $O(1)$.*

Algorithm 5. *Reverse a linked list (existing one, without creating a new one) using as few pointers as possible. Assume you have access to protected head pointer of the list. Time: $O(n)$ and space: $O(1)$.*