# **Delete N Nodes After M Nodes**

**Task:** Given a linked list and two integers M and N. Traverse the linked list such that you retain M nodes then delete next N nodes, continue the same till end of the linked list.

**Examples:**

Input:

M = 2, N = 2

Linked List: 1->2->3->4->5->6->7->8

Output:

Linked List: 1->2->5->6

Input:

M = 3, N = 2

Linked List: 1->2->3->4->5->6->7->8->9->10

Output:

Linked List: 1->2->3->6->7->8

Input:

M = 1, N = 1

Linked List: 1->2->3->4->5->6->7->8->9->10

Output:

Linked List: 1->3->5->7->9

**Notes:**

* The main part of the problem is to maintain proper links between nodes, make sure that all corner cases are handled.
* Assume that M cannot be 0.

**Singles to Doubles**

**Task:** convert a single linked list to a double linked list

# **Reverse the List**

**Task:** Write a function that will reverse a linked list

# **Linked Palindrome**

**Task:** Given a linked list of characters, recursively check if it is a palindrome or not

**Examples:**

Input: A —> B —> C —> B —> A —> null

Output: The linked list is a palindrome

Input: A —> B —> C —> C —> B —> null

Output: The linked list is not a palindrome

**Notes:** This can be done without constructing a string out of the characters

# **Monkey in the Middle**

**Task:** Given a singly linked list, find the middle of the linked list. If there are even nodes, then there would be two middle nodes, we need to print the second middle element.

**Examples:**

Given: 1 -> 2 -> 3 -> 4 -> 5

Output: 3

Given: 1 -> 2- >3 ->4 ->5 -> 6

Output: 4

# **Up and Down Sort**

**Task:** Write two functions that take in the head of a linked list:

**“public ListNode SortListAsc(ListNode head)” “public ListNode SortListDesc(ListNode head)”**

One will return the list sorted in *ascending* order, the other in *descending* order.

**Notes:**

* The number of nodes in the list is in the range [0, 5 \* 10^4].
* -10^5 <= Node.val <= 10^5

# **Nth From the End**

**Task:** Given a linked list consisting of **L** nodes and given a number **N**, complete the function getNthFromLast() which takes two arguments: **reference to the head** and **N** which will return the mode Nth from the end or -1 in case the node doesn't exist.

**Examples:**

Input:

N = 2

Linked List: 1->2->3->4->5->6->7->8->9

Output: 8

**Notes:** Assume that 1<= **L,**N<= 10^3

# **Node Insertion**

**Task:** Given a sorted list **L** in increasing order and a single node **N**, insert the node into the list's correct sorted position. The function should take an existing node and rearranges pointers to insert it into the list.

**Examples:**

Input:

N = 7

Linked List: 3 -> 5 -> 6 -> 9 -> 15 -> 23

Output: 3 -> 5 -> 6-> 7 -> 9 -> 15 -> 23

**Note:** There are many possible solutions to this problem.