**Problem 1:** [**876. Middle of the Linked List**](https://leetcode.com/problems/middle-of-the-linked-list/)

**Given the head of a singly linked list, return *the middle node of the linked list*. If there are two middle nodes, return the second middle node.**

**Example 1:**

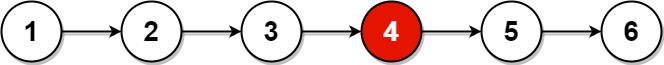
****

**Input: head = [1,2,3,4,5]**

**Output: [3,4,5]**

**Explanation: The middle node of the list is node 3.**

**Example 2:**

****

**Input: head = [1,2,3,4,5,6]**

**Output: [4,5,6]**

**Explanation: Since the list has two middle nodes with values 3 and 4, we return the second one.**

**Constraints:**

* **The number of nodes in the list is in the range [1, 100].**
* **1 <= Node.val <= 100**

**Solution:**



**Or…**



**Problem 2:** [**234. Palindrome Linked List**](https://leetcode.com/problems/palindrome-linked-list/)

**Given the head of a singly linked list, return true*if it is a Palindrome* *or*false*otherwise*.**

**Example 1:**

****

**Input: head = [1,2,2,1]**

**Output: true**

**Example 2:**

****

**Input: head = [1,2]**

**Output: false**

**Constraints:**

* **The number of nodes in the list is in the range [1, 105].**
* **0 <= Node.val <= 9**

**Solution:**

**Or…**

**Or…**



**Problem 3:** [**206. Reverse Linked List**](https://leetcode.com/problems/reverse-linked-list/)

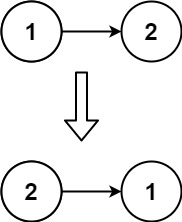
**Given the head of a singly linked list, reverse the list, and return *the reversed list*.**

**Example 1:**

****

**Input: head = [1,2,3,4,5]**

**Output: [5,4,3,2,1]**

**Example 2: **

**Input: head = [1,2]**

**Output: [2,1]**

**Example 3:**

**Input: head = []**

**Output: []**

**Constraints:**

* **The number of nodes in the list is the range [0, 5000].**
* **-5000 <= Node.val <= 5000**

**Solution:**



**Problem 4:** [**160. Intersection of Two Linked Lists**](https://leetcode.com/problems/intersection-of-two-linked-lists/)

**Given the heads of two singly linked-lists headA and headB, return *the node at which the two lists intersect*. If the two linked lists have no intersection at all, return null.**

**For example, the following two linked lists begin to intersect at node c1:**

****

**The test cases are generated such that there are no cycles anywhere in the entire linked structure.**

**Note that the linked lists must retain their original structure after the function returns.**

**Custom Judge:**

**The inputs to the judge are given as follows (your program is not given these inputs):**

* **intersectVal - The value of the node where the intersection occurs. This is 0 if there is no intersected node.**
* **listA - The first linked list.**
* **listB - The second linked list.**
* **skipA - The number of nodes to skip ahead in listA (starting from the head) to get to the intersected node.**
* **skipB - The number of nodes to skip ahead in listB (starting from the head) to get to the intersected node.**

**The judge will then create the linked structure based on these inputs and pass the two heads, headA and headB to your program. If you correctly return the intersected node, then your solution will be accepted.**

**Example 1:**

****

**Output: Intersected at '8'**

**Explanation: The intersected node's value is 8 (note that this must not be 0 if the two lists intersect).**

**From the head of A, it reads as [4,1,8,4,5]. From the head of B, it reads as [5,6,1,8,4,5]. There are 2 nodes before the intersected node in A; There are 3 nodes before the intersected node in B.**

**- Note that the intersected node's value is not 1 because the nodes with value 1 in A and B (2nd node in A and 3rd node in B) are different node references. In other words, they point to two different locations in memory, while the nodes with value 8 in A and B (3rd node in A and 4th node in B) point to the same location in memory.**

**Solution:**



**Problem 5: [147. Insertion Sort List](https://leetcode.com/problems/insertion-sort-list/)**

**Given the head of a singly linked list, sort the list using insertion sort, and return *the sorted list's head*.**

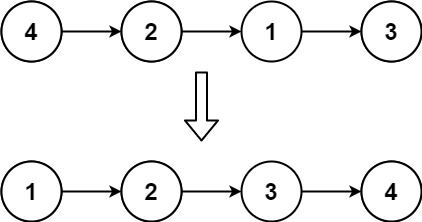
**The steps of the insertion sort algorithm:**

1. **Insertion sort iterates, consuming one input element each repetition and growing a sorted output list.**
2. **At each iteration, insertion sort removes one element from the input data, finds the location it belongs within the sorted list and inserts it there.**
3. **It repeats until no input elements remain.**

**The following is a graphical example of the insertion sort algorithm. The partially sorted list (black) initially contains only the first element in the list. One element (red) is removed from the input data and inserted in-place into the sorted list with each iteration.**

****

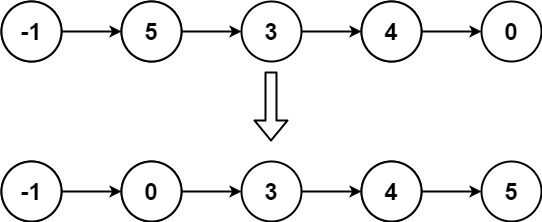
**Example 1:**

****

**Input: head = [4,2,1,3]**

**Output: [1,2,3,4]**

**Example 2:**

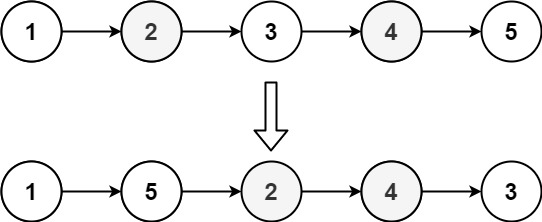
****

**Input: head = [-1,5,3,4,0]**

**Output: [-1,0,3,4,5]**

**Constraints:**

* **The number of nodes in the list is in the range [1, 5000].**
* **-5000 <= Node.val <= 5000**

****

**Problem 5: Reorder linked list.**

**Input: head = [1,2,3,4,5]**

**Output: [1,5,2,4,3]**

**Constraints:**

* **The number of nodes in the list is in the range [1, 5 \* 104].**
* **1 <= Node.val <= 1000**

**Solution:**



**Problem 7:** [**142. Linked List Cycle II**](https://leetcode.com/problems/linked-list-cycle-ii/)

**Given the head of a linked list, return *the node where the cycle begins. If there is no cycle, return*null.**

**There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to (0-indexed). It is -1 if there is no cycle. Note that pos is not passed as a parameter.**

**Do not modify the linked list.**

**Example 1:**

****



**Problem 8:** [**141. Linked List Cycle**](https://leetcode.com/problems/linked-list-cycle/)

**Example 1:**

****

**Input: head = [3,2,0,-4], pos = 1**

**Output: true**

**Explanation: There is a cycle in the linked list, where the tail connects to the 1st node (0-indexed).**

**Example 2:**

**Output: true**

**Explanation: There is a cycle in the linked list, where the tail connects to the 0th node.**

**Example 3:**

****



**Solution:**



**Problem 9: Cycle Removal**

**Solution:**



**Problem 10:** [**21. Merge Two Sorted Lists**](https://leetcode.com/problems/merge-two-sorted-lists/)

**You are given the heads of two sorted linked lists list1 and list2.**

**Merge the two lists into one sorted list. The list should be made by splicing together the nodes of the first two lists.**

**Return *the head of the merged linked list*.**

**Example 1:**

****

**Input: list1 = [1,2,4], list2 = [1,3,4]**

**Output: [1,1,2,3,4,4]**

**Example 2:**

**Input: list1 = [], list2 = []**

**Output: []**

**Example 3:**

**Input: list1 = [], list2 = [0]**

**Output: [0]**

**Constraints:**

* **The number of nodes in both lists is in the range [0, 50].**
* **-100 <= Node.val <= 100**
* **Both list1 and list2 are sorted in non-decreasing order.**

**Solution:**



**Problem 11:** [**19. Remove Nth Node From End of List**](https://leetcode.com/problems/remove-nth-node-from-end-of-list/)

**Given the head of a linked list, remove the nth node from the end of the list and return its head.**

**Example 1:**

****

**Input: head = [1,2,3,4,5], n = 2**

**Output: [1,2,3,5]**

**Example 2:**

**Input: head = [1], n = 1**

**Output: []**

**Example 3:**

**Input: head = [1,2], n = 1**

**Output: [1]**

**Solution:**



**Problem 12:** [**2. Add Two Numbers**](https://leetcode.com/problems/add-two-numbers/)

**You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.**

**You may assume the two numbers do not contain any leading zero, except the number 0 itself.**

**Example 1:**

****

**Input: l1 = [2,4,3], l2 = [5,6,4]**

**Output: [7,0,8]**

**Explanation: 342 + 465 = 807.**

**Example 2:**

**Input: l1 = [0], l2 = [0]**

**Output: [0]**

**Example 3:**

**Input: l1 = [9,9,9,9,9,9,9], l2 = [9,9,9,9]**

**Output: [8,9,9,9,0,0,0,1]**

**Constraints:**

* **The number of nodes in each linked list is in the range [1, 100].**
* **0 <= Node.val <= 9**
* **It is guaranteed that the list represents a number that does not have leading zeros.**

**Solution:**



**Assignment**

**Problem 1: Intersection point of two linked lists**

We are given a linked list. There are Q queries. Each query is represented by a index x which represents the position from which the element has to be deleted. After every query print the linked list.

**Input Format**

First line contains the number of nodes in the linked list n and queries q and then next line contains n space separated integers representing the node values of the linked list and then q lines follow containing the indices to be deleted.

**Constraints**

n < 1000

**Output Format**

Output the linked list after every query. Print the node values in a single line separated by spaces.

**Sample Input**

5 2

1 2 3 4 5

0

1

**Sample Output**

2 3 4 5

2 4 5

**Explanation**

Initially the node at index 0 is deleted and list becomes 2 3 4 5. Then node at index 1 is deleted and the list becomes 2 4 5.

**Solution:**



**Problem 2:** Given a singly linked list delete all nodes which are having a greater value among any of the nodes residing on the right side.

**Input Format**

First line contains **N**, the number of nodes in the linked list.  
Next line contains the N nodes data .

**Constraints**

0<N<10^9

**Output Format**

Print the linked list after deleting the nodes which are having a greater value on right side.

**Sample Input**

8

12 15 10 11 5 6 2 3

**Sample Output**

15 11 6 3

**Explanation**

In the given case 12 is having right node greater than itself i.e. 12<15 so delete it.  
Similarly, 10<11, 5<6 and 2<3 so delete 10, 5 and 2 as well.  
Thus, linked list becomes 15->11->6->3.

**Solution:**



**Problem 3:** Given a **head** to Linked List **L**, write a function to reverse the list taking **k** elements at a time. Assume **k** is a factor of the size of List.

You need not have to create a new list. Just reverse the old one using **head**.

**Input Format**

The first line contains 2 space separated integers **N** and **K**

The next line contains **N** space separated integral elements of the list.

**Constraints**

0 <= N <= 10^6 0 <= K <= 10^6

**Output Format**

**Sample Input**

9 3

9 4 1 7 8 3 2 6 5

**Sample Output**

1 4 9 3 8 7 5 6 2

**Solution:**



**Problem 4:** Merge Sort

Solution:

**Input:** head = [4,2,1,3]

**Output:** [1,2,3,4]



**Problem:** [**83. Remove Duplicates from Sorted List**](https://leetcode.com/problems/remove-duplicates-from-sorted-list/)

**Solution:**

**Input: head = [1,1,2]**

**Output: [1,2]**

**Input: head = [1,1,2,3,3]**

**Output: [1,2,3]**

