1. Count nodes of linked list

Given a singly linked list. The task is to find the length of the linked list, where length is defined as the number of nodes in the linked list.

Example 1:

Input:
LinkedList: 1->2->3->4->5
Output: 5
Explanation: Count of nodes in the linked list is 5, which is its length.
Example 2:

Input:
LinkedList: 2->4->6->7->5->1->0
Output: 7
Explanation: Count of nodes in the linked list is 7. Hence, the output
is 7.

Expected Time Complexity : O(N)
Expected Auxiliary Space : O(1)

Constraints:
1 <= N <= 10^5
1 <= value <= 10^3

2.Find length of Loop

Given a linked list of size N. The task is to complete the function countNodesinLoop() that checks whether a given Linked List contains a loop or not and if the loop is present then return the count of nodes in a loop or else return 0. C is the position of the node to which the last node is connected. If it is 0 then no loop.

Example 1:

Input:
N = 10
value[]={25,14,19,33,10,21,39,90,58,45}
C = 4
Output: 7
Explanation: The loop is 45->33. So

length of loop is 33->10->21->39->
90->58->45 = 7. The number 33 is
connected to the last node to form the
loop because according to the input the
4th node from the beginning(1 based
index) will be connected to the last
node for the loop.
Example 2:

Input:
N = 2
value[] = {1,0}
C = 1
Output: 2
Explanation: The length of the loop
is 2.

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(1)

Constraints:
1 <= N <= 500
0 <= C <= N-1

3.Linked List Insertion

Create a link list of size N according to the given input literals. Each integer input is accompanied by an indicator which can either be 0 or 1. If it is 0, insert the integer in the beginning of the link list. If it is 1, insert the integer at the end of the link list.
Hint: When inserting at the end, make sure that you handle NULL explicitly.

Example 1:
Input:
LinkedList: 9->0->5->1->6->1->2->0->5->0
Output: 5 2 9 5 6
Explanation:
Length of Link List = N = 5
9 0 indicated that 9 should be inserted in the beginning. Modified
Link List = 9.
5 1 indicated that 5 should be inserted in the end. Modified Link
List = 9,5.
6 1 indicated that 6 should be inserted in the end. Modified Link

List = 9,5,6.
2 0 indicated that 2 should be inserted in the beginning. Modified
Link List = 2,9,5,6.
5 0 indicated that 5 should be inserted in the beginning. Modified
Link List = 5,2,9,5,6.
Final linked list = 5, 2, 9, 5, 6.

Example 2:

Input:
LinkedList: 5->1->6->1->9->1
Output: 5 6 9

Expected Time Complexity: O(1) for insertAtBeginning() and O(N) for insertAtEnd().
Expected Auxiliary Space: O(1) for both.

Constraints:
1 <= N <= 10^4

4.Doubly linked list Insertion at given position

Given a doubly-linked list, a position p, and an integer x. The task is to add a new node with value x at the position just after p^{th} node in the doubly linked list.

Example 1:

Input:
LinkedList: 2<->4<->5
p = 2, x = 6
Output: 2 4 5 6
Explanation: p = 2, and x = 6. So, 6 is inserted after p, i.e, at position 3 (0-based indexing).
Example 2:

Input:
LinkedList: 1<->2<->3<->4
p = 0, x = 44
Output: 1 44 2 3 4
Explanation: p = 0, and x = 44 . So, 44 is inserted after p, i.e, at position 1 (0-based indexing).

Expected Time Complexity : O(N)
Expected Auxiliary Space : O(1)

Constraints:
1 <= N <= 10^4
0 <= p < N

5. Reverse a linked list

Given a linked list of N nodes. The task is to reverse this list.

Example 1:

Input:
LinkedList: 1->2->3->4->5->6
Output: 6 5 4 3 2 1
Explanation: After reversing the list, elements are 6->5->4->3->2->1.
Example 2:

Input:
LinkedList: 2->7->8->9->10
Output: 10 9 8 7 2
Explanation: After reversing the list, elements are 10->9->8->7->2.

Expected Time Complexity: O(N).
Expected Auxiliary Space: O(1).

Constraints:
1 <= N <= 10^4

6. Add two numbers represented by Linked List

Given two numbers represented by two linked lists, write a function that returns a Sum list. The sum list is a linked list representation of addition of two input numbers.

Example 1:

Input:
S1 = 3, S2 = 3
ValueS1 = {2,3,4}
ValueS2 = {3,4,5}
Output: 5 7 9
Explanation: After adding the 2 numbers the resultant number is 5 7 9.

Example 2:

Input:
S1 = 1, S2 = 2
ValueS1 = {9}
ValueS2 = {8,7}
Output: 9 6
Explanation: Add 9 and 7 we get 16. 1 is carried here and is added to 8. So the answer is 9 6

Constraints:
1 <= S1, S2 <= 100

7. Swap Kth nodes from ends

Given a singly linked list of size N, and an integer K. You need to swap the Kth node from the beginning and Kth node from the end of the linked list. Swap the nodes through the links. Do not change the content of the nodes.

Example 1:

Input:
N = 4, K = 1
value[] = {1,2,3,4}
Output: 1
Explanation: Here K = 1, hence after swapping the 1st node from the beginning and end the new list will be 4 2 3 1.

Example 2:

Input:
N = 5, K = 7
value[] = {1,2,3,4,5}
Output: 1
Explanation: K > N. Swapping is invalid. Return the head node as it is.

Expected Time Complexity: O(n)
Expected Auxiliary space Complexity: O(1)

Constraints:
1 <= N <= 10^3
1 <= K <= 10^3

8. Delete without head pointer

You are given a pointer/ reference to the node which is to be deleted from the linked list of N nodes. The task is to delete the node. Pointer/ reference to the head node is not given.
Note: No head reference is given to you. It is guaranteed that the node to be deleted is not a tail node in the linked list.

Example 1:

Input:
N = 2
value[] = {1,2}
node = 1
Output: 2
Explanation: After deleting 1 from the linked list, we have remaining nodes as 2.
Example 2:

Input:
N = 4
value[] = {10,20,4,30}
node = 20
Output: 10 4 30
Explanation: After deleting 20 from the linked list, we have remaining nodes as 10, 4 and 30.

Expected Time Complexity : O(1)
Expected Auxiliary Space : O(1)

Constraints:
2 <= N <= 10^3

9. Remove duplicate element from sorted Linked List

Given a singly linked list consisting of N nodes. The task is to remove duplicates (nodes with duplicate values) from the given list (if exists).

Note: Try not to use extra space. The nodes are arranged in a sorted way.

Example 1:

Input:
LinkedList: 2->2->4->5
Output: 2 4 5
Explanation: In the given linked list 2 ->2 -> 4-> 5, only 2 occurs more than 1 time. So we need to remove it once.

Example 2:

Input:
LinkedList: 2->2->2->2->2
Output: 2
Explanation: In the given linked list 2 ->2 ->2 ->2 ->2, 2 is the only element and is repeated 5 times. So we need to remove any four 2.

Expected Time Complexity : O(N)
Expected Auxiliary Space : O(1)

Constraints:
1 <= Number of nodes <= 10^5

10. Identical Linked Lists

Given two Singly Linked List of N and M nodes respectively. The task is to check whether two linked lists are identical or not.
Two Linked Lists are identical when they have the same data and with the same arrangement too.

Example 1:

Input:
LinkedList1: 1->2->3->4->5->6
LinkedList2: 99->59->42->20
Output: Not identical
Example 2:

Input:
LinkedList1: 1->2->3->4->5
LinkedList2: 1->2->3->4->5

Output: Identical

Constraints:
1 <= N <= 10^3

Expected Time Complexity : O(N)
Expected Auxiliary Space : O(1)

11. Insert in Middle of Linked List

Given a linked list of size N and a key. The task is to insert the key in the middle of the linked list.

Example 1:

Input:
LinkedList = 1->2->4
key = 3
Output: 1 2 3 4
Explanation: The new element is inserted after the current middle element in the linked list.
Example 2:

Input:
LinkedList = 10->20->40->50
key = 30
Output: 10 20 30 40 50
Explanation: The new element is inserted after the current middle element in the linked list and Hence, the output is 10 20 30 40 50.

Expected Time Complexity : O(N)
Expected Auxiliary Space : O(1)

Constraints:
1 <= N <= 10^4

12. Merge Sort on Doubly Linked List

Given Pointer/Reference to the head of a doubly linked list of N nodes, the task is to Sort the given doubly linked list using Merge Sort in both non-decreasing and non-increasing order.

Example 1:

Input:
N = 8
value[] = {7,3,5,2,6,4,1,8}
Output:
1 2 3 4 5 6 7 8
8 7 6 5 4 3 2 1
Explanation: After sorting the given linked list in both ways, the resultant matrix will be as given in the first two lines of output, where the first line is the output for non-decreasing order and next line is for non - increasing order.

Example 2:

Input:
N = 5
value[] = {9,15,0,-1,0}
Output:
-1 0 0 9 15
15 9 0 0 -1
Explanation: After sorting the given linked list in both ways, the resultant list will be -1 0 0 9 15 in non-decreasing order and 15 9 0 0 -1 in non-increasing order.

Constraints:
1 <= N <= 10^5

13. Nth node from end of linked list

Given a linked list consisting of L nodes and given a number N. The task is to find the Nth node from the end of the linked list.

Example 1:

Input:
N = 2

LinkedList: 1->2->3->4->5->6->7->8->9
Output: 8
Explanation: In the first example, there are 9 nodes in the linked list and we need to find the 2nd node from the end. the 2^{nd} node from the end is 8.
Example 2:

Input:
N = 5
LinkedList: 10->5->100->5
Output: -1
Explanation: In the second example, there are 4 nodes in the linked list and we need to find 5th from the end. Since 'n' is more than the number of nodes in the linked list, the output is -1.

Note:
Try to solve it in a single traversal.

Expected Time Complexity: O(N).
Expected Auxiliary Space: O(1).

Constraints:
1 <= L <= 10^6
1 <= N <= 10^6

14. Given a linked list of 0s, 1s and 2s, sort it.

Given a linked list of N nodes where nodes can contain values 0s, 1s, and 2s only. The task is to segregate 0s, 1s, and 2s linked lists such that all zeros segregate to the head side, 2s at the end of the linked list, and 1s in the middle of 0s and 2s.

Example 1:

Input:
N = 8
value[] = {1,2,2,1,2,0,2,2}
Output: 0 1 1 2 2 2 2 2
Explanation: All the 0s are segregated to the left end of the linked list, 2s to the right end of the list, and 1s in between.

Example 2:

Input:
N = 4

value[] = {2,2,0,1}
Output: 0 1 2 2
Explanation: After arranging all the 0s,1s and 2s in the given format, the output will be 0 1 2 2.

Expected Time Complexity: O(N).
Expected Auxiliary Space: O(N).

Constraints:
1 <= N <= 10^6

15. Remove duplicates from an unsorted linked list

Given an unsorted linked list of N nodes. The task is to remove duplicate elements from this unsorted Linked List. When a value appears in multiple nodes, the node which appeared first should be kept, all other duplicates are to be removed.

Example 1:

Input:
N = 4
value[] = {5,2,2,4}
Output: 5 2 4
Explanation:Given linked list elements are 5->2->2->4, in which 2 is repeated only. So, we will delete the extra repeated elements 2 from the linked list and the resultant linked list will contain 5->2->4

Example 2:

Input:
N = 5
value[] = {2,2,2,2,2}
Output: 2
Explanation:Given linked list elements are 2->2->2->2->2, in which 2 is repeated. So, we will delete the extra repeated elements 2 from the linked list and the resultant linked list will contain only 2.

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(N)

Constraints:
1 <= size of linked lists <= 10^6

0 <= numbers in list <= 10^4

16. Rotate a Linked List

Given a singly linked list of size N. The task is to left-shift the linked list by k nodes, where k is a given positive integer smaller than or equal to length of the linked list.

Example 1:

Input:
N = 5
value[] = {2, 4, 7, 8, 9}
k = 3
Output: 8 9 2 4 7
Explanation:
Rotate 1: 4 -> 7 -> 8 -> 9 -> 2
Rotate 2: 7 -> 8 -> 9 -> 2 -> 4
Rotate 3: 8 -> 9 -> 2 -> 4 -> 7
Example 2:

Input:
N = 8
value[] = {1, 2, 3, 4, 5, 6, 7, 8}
k = 4
Output: 5 6 7 8 1 2 3 4

Expected Time Complexity: O(N).
Expected Auxiliary Space: O(1).

Constraints:
1 <= N <= 10^3
1 <= k <= N

17. Detect Loop in linked list

Given a linked list of N nodes. The task is to check if the linked list has a loop. Linked list can contain a self loop.

Example 1:

Input:
N = 3
value[] = {1,3,4}
x(position at which tail is connected) = 2
Output: True
Explanation: In above test case N = 3. The linked list with nodes N = 3 is given. Then the value of x=2 is given which means the last node is connected with the x^{th} node of the linked list. Therefore, there exists a loop.
Example 2:

Input:
N = 4
value[] = {1,8,3,4}
x = 0
Output: False
Explanation: For N = 4 ,x = 0 means
the lastNode->next = NULL, then
the Linked list does not contains
any loop.

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(1)

Constraints:
1 ≤ N ≤ 10^4
1 ≤ Data on Node ≤ 10^3

18. Remove loop in Linked List

Given a linked list of N nodes such that it may contain a loop.

A loop here means that the last node of the link list is connected to the node at position X(1-based index). If the link list does not have any loop, X=0.

Remove the loop from the linked list, if it is present, i.e. unlink the last node which is forming the loop.

Example 1:

Input:
N = 3
value[] = {1,3,4}
X = 2
Output: 1
Explanation: The link list looks like
1 -> 3 -> 4
^ |
|____|
A loop is present. If you remove it
successfully, the answer will be 1.

Example 2:

Input:
N = 4
value[] = {1,8,3,4}
X = 0
Output: 1
Explanation: The Linked list does not
contains any loop.

Example 3:

Input:
N = 4
value[] = {1,2,3,4}
X = 1
Output: 1
Explanation: The link list looks like
1 -> 2 -> 3 -> 4
^ |
|_____|
A loop is present.
If you remove it successfully,
the answer will be 1.

Expected time complexity: O(N)
Expected auxiliary space: O(1)

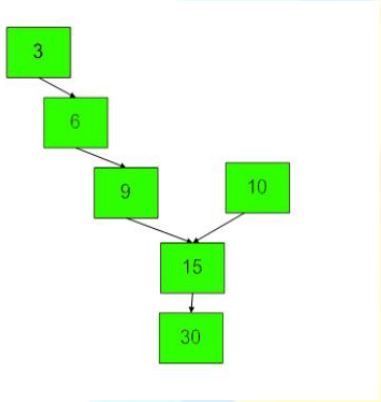
Constraints:
1 ≤ N ≤ 10^4

19. Intersection Point in Y Shaped Linked Lists

Given two singly linked lists of size N and M, write a program to get the point where two linked lists intersect each other.

Example 1:

Input:
LinkList1 = 3->6->9->common
LinkList2 = 10->common
common = 15->30->NULL
Output: 15
Explanation:



Example 2:

Input:
Linked List 1 = 4->1->common
Linked List 2 = 5->6->1->common
common = 8->4->5->NULL
Output: 8
Explanation:

```
4              5
|            |
1            6
 \          /
  8   -----  1
   |
   4
   |
  5
  |
```

NULL

Expected Time Complexity: O(N+M)
Expected Auxiliary Space: O(1)

Constraints:
1 ≤ N + M ≤ 2*105
-1000 ≤ value ≤ 1000

20. LRU Cache

Design a data structure that works like a LRU Cache. Here cap denotes the capacity of the cache and Q denotes the number of queries. Query can be of two types:

SET x y: sets the value of the key x with value y
GET x: gets the key of x if present else returns -1.

The LRUCache class has two methods get() and set() which are defined as follows.

get(key): returns the value of the key if it already exists in the cache otherwise returns -1.
set(key, value): if the key is already present, update its value. If not present, add the key-value pair to the cache. If the cache reaches its capacity it should invalidate the least recently used item before inserting the new item.
In the constructor of the class the capacity of the cache should be initialized.

Example 1:

Input:
cap = 2
Q = 2
Queries = SET 1 2 GET 1
Output: 2
Explanation:
Cache Size = 2

SET 1 2 GET 1
SET 1 2 : 1 -> 2

GET 1 : Print the value corresponding

to Key 1, ie 2.

Example 2:

Input:
cap = 2
Q = 8
Queries = SET 1 2 SET 2 3 SET 1 5
SET 4 5 SET 6 7 GET 4 SET 1 2 GET 3
Output: 5 -1
Explanation:
Cache Size = 2
SET 1 2 : 1 -> 2

SET 2 3 : 1 -> 2, 2 -> 3 (the most recently
used one is kept at the rightmost position)

SET 1 5 : 2 -> 3, 1 -> 5

SET 4 5 : 1 -> 5, 4 -> 5 (Cache size is 2, hence
we delete the least recently used key-value pair)

SET 6 7 : 4 -> 5, 6 -> 7

GET 4 : Prints 5 (The cache now looks like
6 -> 7, 4->5)

SET 1 2 : 4 -> 5, 1 -> 2
(Cache size is 2, hence we delete the least
recently used key-value pair)

GET 3 : No key value pair having
key = 3. Hence, -1 is printed.

Expected Time Complexity: O(1) for both get() and set().
Expected Auxiliary Space: O(1) for both get() and set().
(Although, you may use extra space for cache storage and implementation purposes).

Constraints:
1 <= cap <= 10^3
1 <= Q <= 10^5
1 <= x, y <= 10^4

21. Merge two sorted linked lists

Given two sorted linked lists consisting of N and M nodes respectively. The task is to merge both of the list (in-place) and return head of the merged list.

Example 1:

Input:
N = 4, M = 3
valueN[] = {5,10,15,40}
valueM[] = {2,3,20}
Output: 2 3 5 10 15 20 40
Explanation: After merging the two linked lists, we have merged list as 2, 3, 5, 10, 15, 20, 40.
Example 2:

Input:
N = 2, M = 2
valueN[] = {1,1}
valueM[] = {2,4}
Output:1 1 2 4
Explanation: After merging the given two linked lists , we have 1, 1, 2, 4 as output.

Expected Time Complexity : O(n+m)
Expected Auxiliary Space : O(1)

Constraints:
1 <= N, M <= 104
0 <= Node's data <= 105

22. Merge K sorted linked lists

Given K sorted linked lists of different sizes. The task is to merge them in such a way that after merging they will be a single sorted linked list.

Example 1:

Input:
K = 4
value = {{1,2,3},{4 5},{5 6},{7,8}}
Output: 1 2 3 4 5 5 6 7 8

Explanation:
The test case has 4 sorted linked
list of size 3, 2, 2, 2
1st list 1 -> 2-> 3
2nd list 4->5
3rd list 5->6
4th list 7->8
The merged list will be
1->2->3->4->5->5->6->7->8.
Example 2:

Input:
K = 3
value = {{1,3},{4,5,6},{8}}
Output: 1 3 4 5 6 8
Explanation:
The test case has 3 sorted linked
list of size 2, 3, 1.
1st list 1 -> 3
2nd list 4 -> 5 -> 6
3rd list 8
The merged list will be
1->3->4->5->6->8.
Expected Time Complexity: O(nk Logk)
Expected Auxiliary Space: O(k)
Note: n is the maximum size of all the k link list

Constraints
1 <= K <= 103

23. Merge Sort for Linked List

Given Pointer/Reference to the head of the linked list, the task is to Sort the given linked list using Merge Sort.
Note: If the length of the linked list is odd, then the extra node should go in the first list while splitting.

Example 1:

Input:
N = 5
value[] = {3,5,2,4,1}

Output: 1 2 3 4 5
Explanation: After sorting the given linked list, the resultant matrix will be 1->2->3->4->5.
Example 2:

Input:
N = 3
value[] = {9,15,0}
Output: 0 9 15
Explanation: After sorting the given linked list , the resultant will be 0->9->15.

Expected Time Complexity: O(N*Log(N))
Expected Auxiliary Space: O(N)

Constraints:
1 <= N <= 105

24. Check if Linked List is Palindrome

Given a singly linked list of size N of integers. The task is to check if the given linked list is palindrome or not.

Example 1:

Input:
N = 3
value[] = {1,2,1}
Output: 1
Explanation: The given linked list is
1 2 1 , which is a palindrome and
Hence, the output is 1.

Example 2:

Input:
N = 4
value[] = {1,2,3,4}
Output: 0
Explanation: The given linked list
is 1 2 3 4 , which is not a palindrome
and Hence, the output is 0.

Expected Time Complexity: O(N)
Expected Auxiliary Space Usage: O(1) (ie, you should not use the recursive stack space as well)

Constraints:
1 <= N <= 10^5

25. Finding middle element in a linked list

Given a singly linked list of N nodes.
The task is to find the middle of the linked list. For example, if the linked list is
1-> 2->3->4->5, then the middle node of the list is 3.
If there are two middle nodes(in case, when N is even), print the second middle element.
For example, if the linked list given is 1->2->3->4->5->6, then the middle node of the list is 4.

Example 1:

Input:
LinkedList: 1->2->3->4->5
Output: 3
Explanation: Middle of the linked list is 3.
Example 2:

Input:
LinkedList: 2->4->6->7->5->1
Output: 7
Explanation:
Middle of the linked list is 7.

Expected Time Complexity: O(N).
Expected Auxiliary Space: O(1).

Constraints:
1 <= N <= 5000