Problems on LL – Set 2

1.Delete nodes which have a greater value on right side

Given a singly linked list, remove all the nodes which have a greater value on right side.

Examples:
a) The list 12->15->10->11->5->6->2->3->NULL should be changed to 15->11->6->3->NULL. Note that 12, 10, 5 and 2 have been deleted because there is a greater value on the right side. When we examine 12, we see that after 12 there is one node with value greater than 12 (i.e. 15), so we delete 12. When we examine 15, we find no node after 15 that has value greater than 15 so we keep this node. When we go like this, we get 15->6->3

b) The list 10->20->30->40->50->60->NULL should be changed to 60->NULL. Note that 10, 20, 30, 40 and 50 have been deleted because they all have a greater value on the right side. **c)** The list 60->50->40->30->20->10->NULL should not be changed

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

Example: Input: List1: 5->6->3 // represents number 365

List2: 8->4->2 // represents number 248

Output: Resultant list: 3->1->6 // represents number 613

List2: 8->4 // represents number 48

Output: Resultant list: 5->0->0->5->6 // represents number 65005

3. Find a triplet from three linked lists with sum equal to a given number

Given three linked lists, say a, b and c, find one node from each list such that the sum of the values of the nodes is equal to a given number. For example, if the three linked lists are 12->6->29, 23->5->8 and 90->20->59, and the given number is 101, the output should be tripel “6 5 90”.

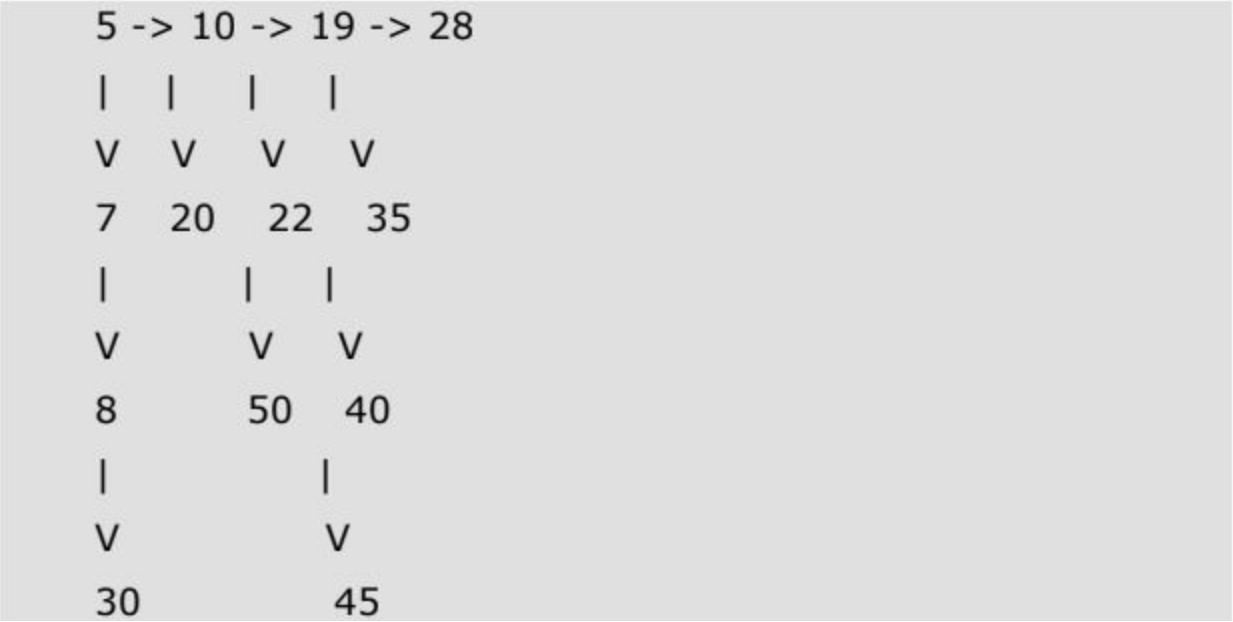
4. Rotate a Linked List

Given a singly linked list, rotate the linked list counter-clockwise by k nodes. Where k is a given positive integer. For example, if the given linked list is 10->20->30->40->50->60 and k is 4, the list should be modified to 50->60->10->20->30->40. Assume that k is smaller than the count of nodes in linked list.

5. Flattening a Linked List

Given a linked list where every node represents a linked list and contains two pointers of its type: (i) Pointer to next node in the main list (we call it ‘right’ pointer in below code)

(ii) Pointer to a linked list where this node is head (we call it ‘down’ pointer in below code). All linked lists are sorted. See the following example



Write a function flatten() to flatten the lists into a single linked list. The flattened linked list should also be sorted. For example, for the above input list, output list should be 5->7->8->10->19->20->22->28->30->35->40->45->50.

6.Given two numbers represented by two linked lists, write a function that returns the sum list. The sum list is linked list representation of the addition of two input numbers. It is not allowed to modify the lists. Also, not allowed to use explicit extra space (Hint: Use Recursion).

Example

Input:

First List: 5->6->3 // represents number 563

Second List: 8->4->2 // represents number 842

Output

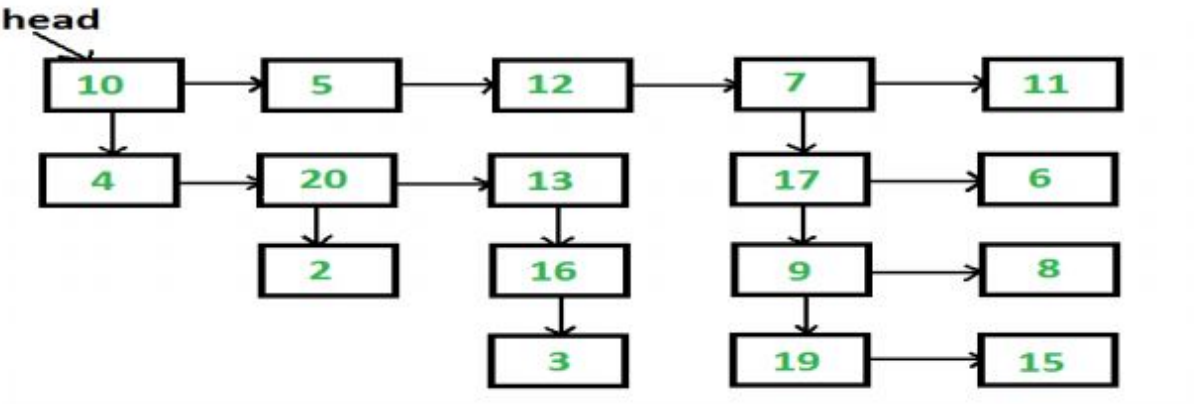
Resultant list: 1->4->0->5 // represents number 1405

7.Sort a linked list of 0s, 1s and 2s

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

Examples: ***Input:*** *1 -> 1 -> 2 -> 0 -> 2 -> 0 -> 1 -> NULL*
Output: *0 -> 0 -> 1 -> 1 -> 1 -> 2 -> 2 -> NULL*
Input: *1 -> 1 -> 2 -> 1 -> 0 -> NULL*
Output: *0 -> 1 -> 1 -> 1 -> 2 -> NULL*

8.Given a linked list where in addition to the next pointer, each node has a child pointer, which may or may not point to a separate list. These child lists may have one or more children of their own, and so on, to produce a multilevel data structure, as shown in below figure.You are given the head of the first level of the list. Flatten the list so that all the nodes appear in a single- level linked list. You need to flatten the list in way that all nodes at first level should come first, then nodes of second level, and so on.



The above list should be converted to 10->5->12->7->11->4->20->13->17->6->2->16->9->8->3->19->15

9.Delete N nodes after M nodes of a linked list

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

10.Pairwise swap elements of a given linked list by changing links

Given a singly linked list, write a function to swap elements pairwise. For example, if the linked list is 1->2->3->4->5->6- >7 then the function should change it to 2->1->4->3->6->5->7, and if the linked list is 1->2->3->4->5->6 then the function should change it to 2->1->4->3->6->5

11.Given a linked list of line segments, remove middle points

Given a linked list of co-ordinates where adjacent points either form a vertical line or a horizontal line. Delete points from the linked list which are in the middle of a horizontal or vertical line.

Examples:

```
Input:  (0,10)->(1,10)->(5,10)->(7,10)
                          |
                        (7,5)->(20,5)->(40,5)
Output: Linked List should be changed to following
      (0,10)->(7,10)
                |
              (7,5)->(40,5)
The given linked list represents a horizontal line from (0,10)
to (7, 10) followed by a vertical line from (7, 10) to (7, 5),
followed by a horizontal line from (7, 5) to (40, 5).
```

Input: (2,3)->(4,3)->(6,3)->(10,3)->(12,3)

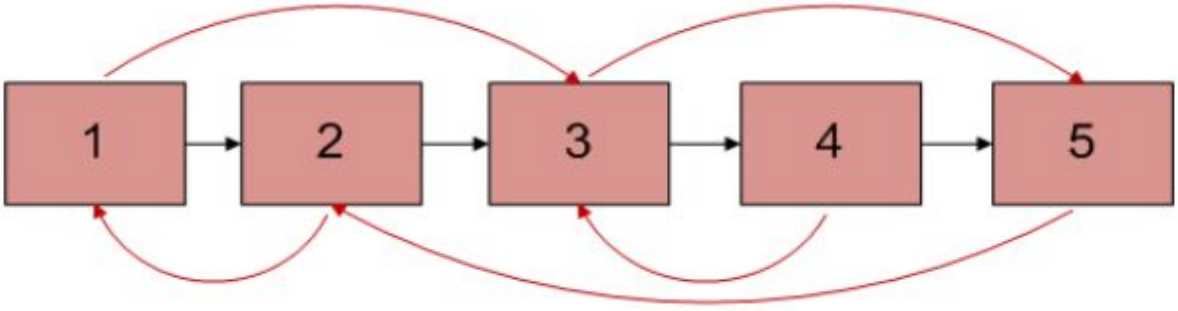
Output: Linked List should be changed to following

(2,3)->(12,3)

There is only one vertical line, so all middle points are removed.

12.Clone a linked list with next and random pointer

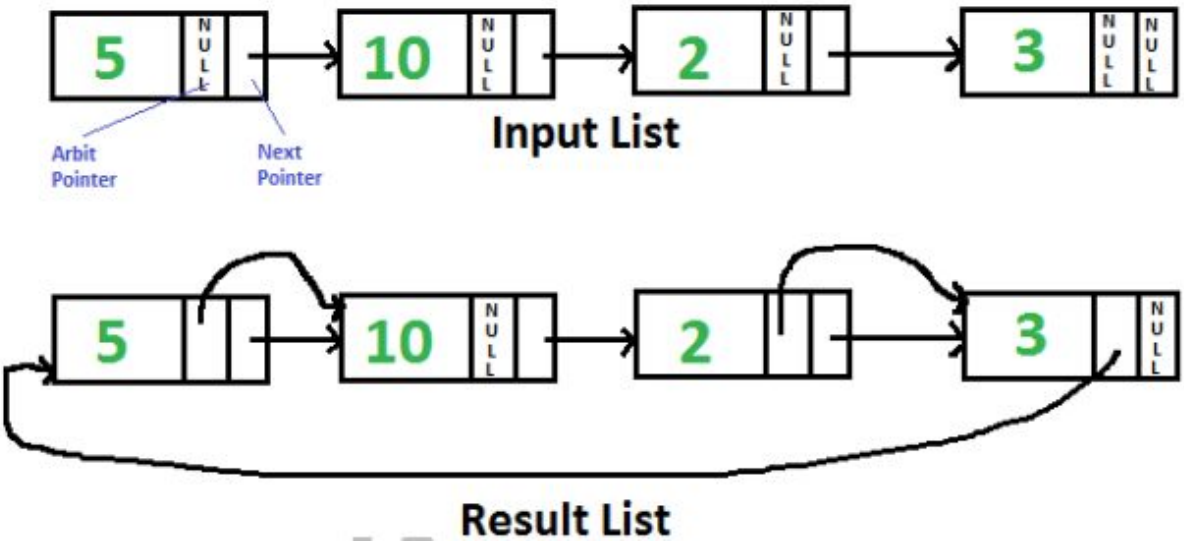
You are given a Double Link List with one pointer of each node pointing to the next node just like in a single link list. The second pointer however CAN point to any node in the list and not just the previous node. Now write a program in **O(n) time** to duplicate this list. That is, write a program which will create a copy of this list. Let us call the second pointer as arbit pointer as it can point to any arbitrary node in the linked list.



13.Insertion Sort for Singly Linked List

14.Point to next higher value node in a linked list with an arbitrary pointer

Given singly linked list with every node having an additional “arbitrary” pointer that currently points to NULL. Need to make the “arbitrary” pointer point to the next higher value node.



15.Rearrange a given linked list in-place.

Given a singly linked list L_0 -> L_1 -> ... -> L_{n-1} -> L_n. Rearrange the nodes in the list so that the new formed list is : L_0 -> L_n -> L_1 -> L_{n-1} -> L_2 -> L_{n-2} ... You are required to do this in-place without altering the nodes' values.
Examples:
Input: 1 -> 2 -> 3 -> 4

Output: 1 -> 4 -> 2 -> 3

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

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

16.Sort a linked list that is sorted alternating ascending and descending orders?

Given a Linked List. The Linked List is in alternating ascending and descending orders. Sort the list efficiently.

Example: Input List: **10**->40->**53**->30->**67**->12->**89**->NULL
Output List: 10->12->30->43->53->67->89->NULL

17.Select a Random Node from a Singly Linked List

Given a singly linked list, select a random node from linked list (the probability of picking a node should be 1/N if there are N nodes in list). You are given a random number generator.

18.Compare two strings represented as linked lists

Given two linked lists, represented as linked lists (every character is a node in a linked list). Write a function compare() that works similar to strcmp(), i.e., it returns 0 if both strings are same, 1 if first linked list is lexicographically greater, and -1 if the second string is lexicographically greater.

Examples:
Input: list1 = g->e->e->k->s->a

list2 = g->e->e->k->s->b

Output: -1

Input: list1 = g->e->e->k->s->a

list2 = g->e->e->k->s

Output: 1

Input: list1 = g->e->e->k->s

list2 = g->e->e->k->s

Output: 0