Write functions for the following :

L1 , L2 are pointers to linked list of sorted (ascending order) unique numbers.

L3 is a pointer to linked list of numbers.

1. union function for L3 = union of L1 , L2 (L3 will also be a sorted list)

2. intersection function for L3 = intersection of L1 , L2 (L3 will also be a sorted list)

3. diff function for L3 = difference of L1 , L2 (L3 will also be a sorted list

L1 , L2 are pointers to linked list of Unsorted unique numbers.

L3 is a pointer to linked list of numbers.

4. u-union function for L3 = union of L1 , L2 (data values of L1 should appear first)

5. u-intersection unction for L3 = intersection of L1 , L2 (data values of L1 should appear first)

6, u-diff function for L3 = difference of L1 , L2 : L1 - L2

1. Check if a given LL is a palindrome.

.2. Remove duplicates from a given sorted LL (recursive)

3. Remove duplicates from a given unsorted LL.

4. Delete last occurrence of a duplicate data value node from a given LL (duplicate data values can be there)

Example : input : L = { 1 , 3, 2, 4, 5, 7, 8, 3, 4 , 2, 7, 4, 2 }

output : L = { 1 , 3, 2, 4, 5, 7, 8 , 4 , 2 }

5. Segregate even data values to left and odd data values right of a given LL

input : L = { 4 , 7 , 8, 2, 5, 4, 9, 1, 6 }

output : L = { 4, 8, 2, 4, 6, 7 5 9 1 }

6. insertion sort on a give LL with unique data values

7. reverse a given LL (try for recursion)

8. Swap Kth node from the beginning with Kth node from end in a given LL

input : L = { 4, 9 , 8, 7 , 2 , 6, 1, 5 , 3 }

k = 3 .

output : L = { 4, 9, 1, 7, 2, 6, 8, 5, 3 }

9. Check whether L2 is present in L1.

Example : L1 = { 5, 8, 2, 9, 5, 8, 2, 3, 7, 1 , 6}

L2 = { 8, 2, 3 } output : 1   if L2 = { 8, 2, 7 }   output : 0
if L2 = { 7 , 1} output :

10. Given two linked lists, merge their nodes together to make one list, taking nodes alternatively between the two lists. If either list runs out, all the nodes should be taken from the other list.

Example :

Input : L1 = { 1 , 3, 5 } L2 = { 2, 4, 6, 8, 9}

Output : L3 = { 1, 2, 3, 4, 5, 6, 8, 9 }

Polynomial Addition

Let polynomial : P1 = 12 x7 + 10 x4 + 18 x2

Let polynomial : P2 = 8 x6 + 9 x4 + 27 x + 45

Write code for : P3 = P1 + P2 = 12 x7 + 8 x6+ 19 x4 + 18 x2 + 27 x + 45

Test case Input : (cin >>) : 12 7 10 4 18 2 -1

(cin >>) : 8 6 9 4 27 1 45 0 -1

Output: (cout <<) : 12 7 8 6 19 4 18 2 27 1 45 0

Create a Generic Linked list

whose node value can be either an integer or character.

Read the input as given in the sequence.

If the input is a character, add to the beginning of the list and if it is an integer add at the end.

Input sequence would be : N I 21 T 9 20 C S 45 E

test case input is : 0 N 0 I 1 21 0 T 1 9 1 20 0 C 0 S 1 45 0 E 0 #

(you read using one cin>> for tag , one cin >> for value, till termination character)

The list should be formed with node contents in sequence as:

{ N I T C S E 21 9 20 45 }

Print (cout <<) the contents of the list as output

(test case output): N I T C S E 21 9 20 45

Example(test case) :   (cin >>)   L   =   1 2 3 4 5 6 7 8 -1

Start position   (cin >>)   i = 2

End position   (cin >>)   j = 5

Output (test case) : (cout <<)   1 5 4 3 2 6 7 8