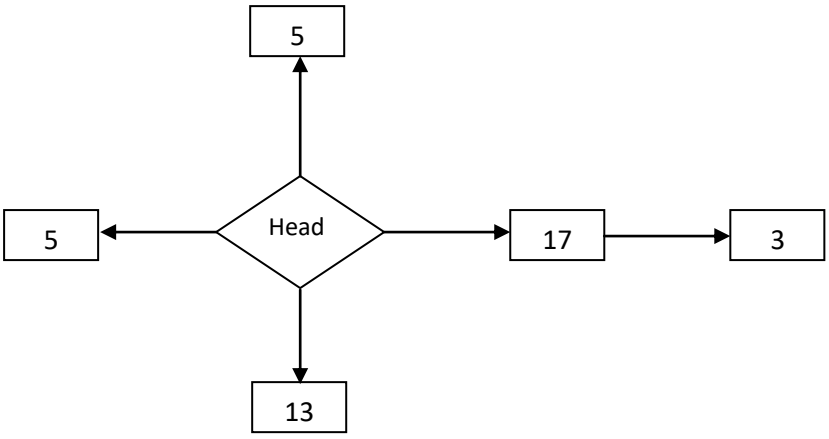
Problem: A

A class **octopus** has a **dummy** head which has four pointers – **left, right, top** and **bottom**. You have to build link list in these 4 directions depending on the input commands. There will be four types of commands – **L, R, T** and **B**. Their specifications are given below:

L x: insert x in the left link list // Where **x** is an integer number.
R x: insert x in the right link list
T x: insert x in the top link list
B x: insert x in the bottom link list

Input: Test case will start with a number **n** indicating the number of lines following it.
Output: you have to give output that which link list (Left, Right, Top and Bottom) has maximum sum of values. See the sample input and output.



Sample input	Sample output
14 L 3 L 15 L 1 R 17 T 9 T 10 B 13 T 5 L 8 R 3 R 12 B 2 B 3 B 4	Right Link List Has Maximum Sum 32

Problem: B

Implement a ***doubly circular SORTED linked list (using dummy node)*** that consists of following operations:

a. Insert (into the sorted position)
b. Show all
c. Search(search for a specific integer data)
d. Delete (find it and delete)
e. Length (current length of the list)

Problem: C

Implement **doubly linked list using dummy node** with following operations.

1. **ins key_of_y key_of_x** (insert command)
   /*
   Insert node x after node y. First search node y using a key value. If node y is found then insert node x after node y. Otherwise insert node x after dummy node.
   */
2. **del key_of_x** (remove command)
   /*
   Search a node x using a key value then delete it from the list if found.
   *
3. **sch search_key** (search command)
   /*
   Search whole linked list against a key number.
   */
4. **shw** (showall command)
   /*
   Traverse whole linked list and print all key values.
   */
5. **ext** (exit command)
   /*
   Exit from the program.
   */

Sample input	**Sample output**
ins 3 1	INSERT after dummy node.
ins 1 2	INSERT after 1.
ins 1 3	INSERT after 1.
ins 2 4	INSERT after 2.
ins 5 0	INSERT after dummy node.
ins 0 2	INSERT after 0.
shw	0 2 1 3 2 4
del 2	Node with key value 2 is DELETED.
del 2	Node with key value 2 is DELETED.
del 2	DELETE not possible.
del 0	Node with key value 0 is DELETED.
shw	1 3 4
sch 1	Node with key value 1 is FOUNDED.
sch 2	Not FOUND.
ext	

Problem: D

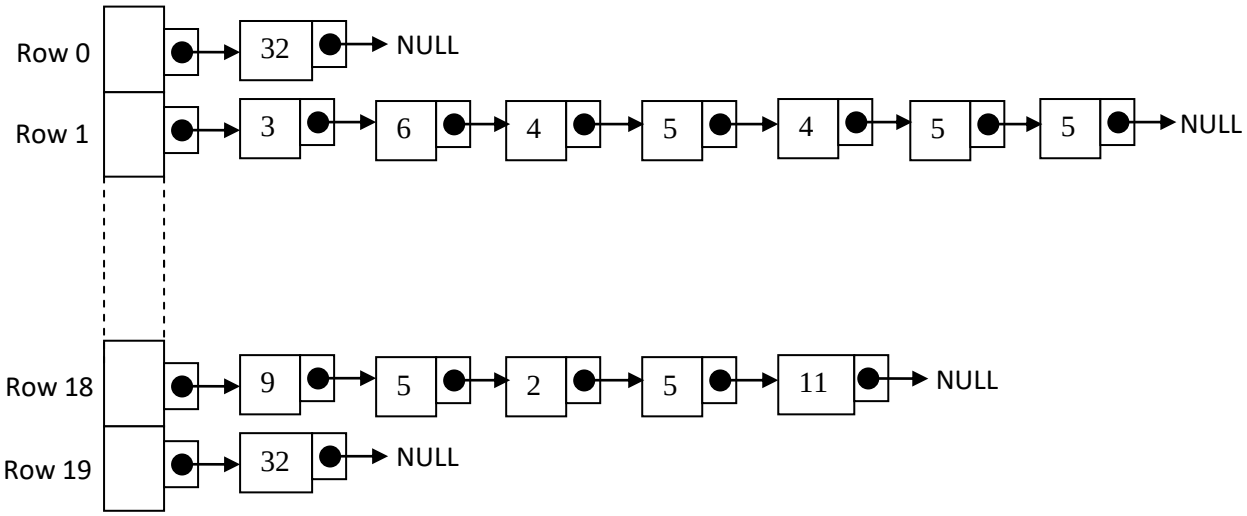
Mirroring Binary Image

Background:

You have given a binary image consist of '0' and '.' where '0' means a black pixel and '.' means a white pixel. Store this image without using any 2D array. Finally show its mirror image. Each row of the image is start and end with at least one black pixel.

Hints:

- Use the concept of 'one singly list per row'
- Each node may contain an integer counter for counting number of consecutive '0's or '.'s.
- Simply insert a node after dummy node of a specific row.

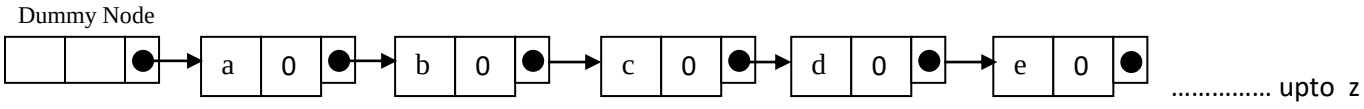


Sample input	***Sample output***
20 32 00000000000000000000000000000000 00000.....0000.....0000......000 0000.00000000.00000.000.00000000 000.000000000.000000000.00000000 000.0000000000.....0000....00000 000.000000000000000.000.00000000 0000.00000000.00000.000.00000000 00000.....0000.....0000......000 00000000000000000000000000000000 00000000000000000000000000000000 0000000000000.000000.00000000000 000000000000..00000..00000000000 00000000000...0000...00000000000 0000000000000.000000.00000000000 0000000000000.000000.00000000000 0000000000000.000000.00000000000 0000000000000.000000.00000000000 000000000000...0000...0000000000 00000000000.....00.....000000000 00000000000000000000000000000000	00000000000000000000000000000000 000......0000.....0000.....00000 00000000.000.00000.00000000.0000 00000000.000000000.000000000.000 00000....0000.....0000000000.000 00000000.000.000000000000000.000 00000000.000.00000.00000000.0000 000......0000.....0000.....00000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000.000000.0000000000000 00000000000..00000..000000000000 00000000000...0000...00000000000 00000000000.000000.0000000000000 00000000000.000000.0000000000000 00000000000.000000.0000000000000 00000000000.000000.0000000000000 0000000000...0000...000000000000 000000000.....00.....00000000000 00000000000000000000000000000000

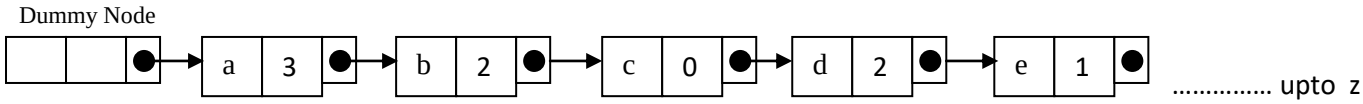
Problem E:

Implement the following sequence of operations one by one

1. Make a linked list for 26 (a to z) english alphabets where each node consists a single alphabet, an integer data for frequency count (initially 0) and next pointer. This list should be sorted in ascending order according to the ASCII value of the alphabets.



2. Now you have given a string consists of english alphabets. Your task is to update the frequency count values of above list for each occurrence of alphabets inside the given string.
   For example, if the string is "babaaedd" then the updated list should be like this:



3. Finally traverse the whole list and print each alphbets with their frequency one by one from a to z. Please follow the sample input-output format. Input string length is not more than 100.

Sample input	Sample output
Mistcsedepartment	a : 1 b : 0 c : 1 d : 1 e : 3 f : 0 g : 0 h : 0 i : 1 j : 0 k : 0 l : 0 m : 2 n : 1 o : 0 p : 1 q : 0 r : 1 s : 2 t : 3 u : 0 v : 0 w : 0 x : 0 y : 0 z : 0

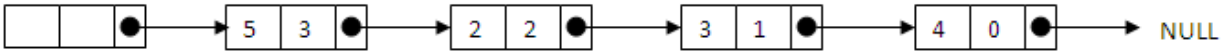
Problem F:

Store two polynomial using single linked lists. Perform addition operation on them and store the output in another single linked list. Define your own class and required functions.

See the following sample input and output.

Sample:
Input:

$5x^3+2x^2+3x+4$



$4x^2+3$

Output:
$5x^3+6x^2+3x+7$