Indian Institute of technology, Guwahati
Department of Computer Science and Engineering
Data Structure Lab (CS210) Assignment: 1

Date: 30th July, 2017. **Total Marks: 20**

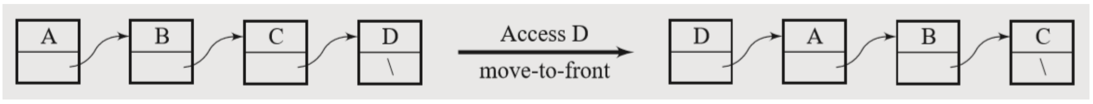
Bob is given by **online stream of alphabets**, for each alphabet he has to search in a list. If the present alphabet is not present in the list, he has to insert the at the end of the list and for every search/insertion he has to count the number of comparisons made. Bob has a restriction that he has to use **linked list** to maintain the list of alphabets.

Bob knows that linked list takes O(n) search time complexity in worst case, so to do this efficiently, he searched over the internet and found a needful information to solve it. But Bob doesn't like to work on this now as he has some other tasks, but he likes to share the information with you, please help him.

Bob's information:
We can improve the efficiency of search by dynamically organizing the list in certain manner, these are called as Self-Organizing lists. This organization depends on the configuration of data; thus, the stream of data requires reorganizing the nodes already on the list. Here we introduce two ways of organizing the lists,

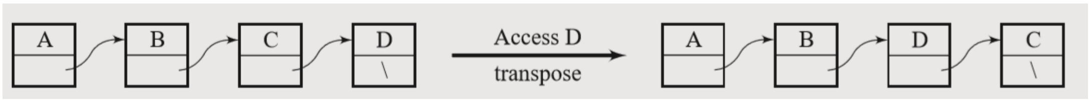
1) Move-to-front Method: If desired element is found, then put it at the beginning of the list, otherwise put it at the end of the list.

   Ex. In the below example, we are searching for D, as it is present in the list, it is moved to the beginning of the list.



2) Transpose Method: If desired element is found, then swap it with its predecessor unless it is at the head of the list, otherwise put it at the end of the list.

   Ex. In the below example, we are searching for D, as it is present in the list, it is swapped with its predecessor.



Note: You have to maintain two different linked lists for two different methods.

Output: For each method,

(1) Print the final order of the alphabets in the linked list after execution of all operations.
(2) Total number of comparisons.

Evaluation Guidelines:

1. 10 marks for transpose and 10 marks for move to front method. Full marks if both methods work for all test cases.
2. If code is not working but most of code is written, maximum 30% can be given based on TA's evaluation.

3. 20% marks will be deducted for each test not running.
4. 10% marks will be deducted for bad coding style. i.e., (1) code is not modular (2) code is not properly indented (3) code is not properly commented and (4) Variable and functions are not suitably named.
5. TA will help you initially to get rid of segmentation fault and compilation error, etc. in your code. TA will not help you find out the solution of the assignment given.

SAMPLE CASES:

INPUT1:

A C B C D A D

OUTPUT:

```
MoveToFront    D A C B        14              //Final list and Total comparisons

Transpose      A C D B        14
```

Explanation:

Input	Move_To_Front	comparisons	Transpose	comparisons
A	A	0	A	0
C	A C	1	A C	1
B	A C B	2	A C B	2
C	C A B	2	C A B	2
D	C A B D	3	C A B D	3
A	A C B D	2	A C B D	2
D	D A C B	4	A C D B	4

INPUT2:

A C B B B

OUTPUT:

```
MoveToFront    B A C          7

Transpose      B A C          8
```

Explanation:

Input	Move_To_Front	comparisons	Transpose	comparisons
A	A	0	A	0
C	A C	1	A C	1
B	A C B	2	A C B	2
B	B A C	3	A B C	3
B	B A C	1	B A C	2