1) Assume header points to a list of integers unless otherwise stated. Make sure your method handles boundary cases. (null list, single nodelist, not found in list etc)

1. addAfter(int b, int data); Write a method that adds a new node containing data after a node containing the value b.

2. delete(int data) : Write a method that deletes a node containing data.

3. replace (int b, int data) ; Write a method that replaces node that contains b with data.

4. count(int data) : Should return the number of times data occurs in the list. The SLL may not be sorted.

5 . boolean issorted() : Write a method that takes a list of integers and returns a boolean stating whether the list is sorted or not.

6. Write a RemoveDuplicates() function which takes a list sorted in increasing order and deletes any duplicate nodes from the list. Ideally, the list should only be traversed once.

7.. AlternatingSplit() : Write a method which takes one list and divides up its nodes to make two smaller lists. The sublists should be made from alternating elements in the orginal list. For eg. if the original list is {a,b,c,d,e}, then one sublist should be {a,c,e} and other other should be {b,d}

8. Given two DLLs, merge their nodes together to make one doubly linked list, taking nodes alternately between the two lists. So ShuffleMerge() with {1, 2, 3} and {7, 13, 1} should yield {1, 7, 2, 13, 3, 1}. If either list runs out, all the nodes should be taken from the other list.

9. void sortedInsertDLL(int data) : Given a sorted DLL of integers, write a method that inserts a new node at the proper place in the DLL. Assume HNODE and TNODE are the sentinel nodes of the DLL.

2) Implement a Text Editor using singly linked lists, doubly linked list or circular lists as data structure

3) Implement a symbol table of an Assembler