Partition List

Given a linked list and a value x, partition it such that all nodes less than x come before nodes greater than or equal to x. You should preserve the original relative order of the nodes in each of the two partitions.

Example:
Input: head = 1->4->3->2->5->2, x = 3
Output: 1->2->2->4->3->5

Remove Duplicates from Sorted List I

Given a sorted linked list, delete all duplicates such that each element appear only once.
Example 1:
Input: 1->1->2
Output: 1->2

Example 2:
Input: 1->1->2->3->3
Output: 1->2->3

Remove Duplicates from Sorted List II

Given a sorted linked list, delete all nodes that have duplicate numbers, leaving only distinct numbers from the original list.
Example 1:
Input: 1->2->3->3->4->4->5
Output: 1->2->5

Example 2:
Input: 1->1->1->2->3
Output: 2->3

Implement and test your code in the SinglyLinkedList.java as flowing:

```
public class SinglyLinkedList {
     // reference that points to the list head
     public ListNode head;

     // nested class for singly-list node
     private static class ListNode {
          int val;
          ListNode next;
          ListNode(int x) {
               val = x;
          }
          ListNode(int x, ListNode nextIn) {
               this.val = x;
               this.next = nextIn;
          }
     }

     public SinglyLinkedList() {
          head = null;
     }

     // add node to the end of list
     private void add(int val) {
          ListNode e = new ListNode(val, head);
          head = e;

     }

     public String toString() {
          String mylist = "";
          ListNode e = head;
          while(e != null) {
               mylist = mylist + e.val + " ";
               e = e.next;
          }
          return mylist;
     }

      public void partition(int x) {
            // place your code here
      }
    public void deleteDuplicates1() {
```

```
        // place your code here
    }

    public  void deleteDuplicates2() {
        // place your code here
    }

     public static void main(String args[]) {
          SinglyLinkedList list1 = new SinglyLinkedList();
          SinglyLinkedList list2 = new SinglyLinkedList();
          SinglyLinkedList list3 = new SinglyLinkedList();
          int[] array1 = {1,4,3,2,5,2};
          int[] array2 = {1,1,2,3,3,3};
          int[] array3 = {1,1,2,2,2,3};
          for(int i = 5; i > -1; i--) {
               list1.add(array1[i]);
               list2.add(array2[i]);
              list3.add(array3[i]);
          }
          System.out.println(list1);
          list1.partition(3);
          System.out.println(list1);

          System.out.println(list2);
          list2.deleteDuplicates1();
          System.out.println(list2);

          System.out.println(list3);
          list3.deleteDuplicates2();
          System.out.println(list3);
     }
}
```

The expected output of the code is as follows:

```
1 4 3 2 5 2
1 2 2 4 3 5
1 1 2 3 3 3
1 2 3
1 1 2 2 2 3
3
```