Remove duplicates from a sorted Singly Linked List

Write a function that takes a list sorted in non-decreasing order and deletes any duplicate nodes from the list. The list should only be traversed once.
For example if the linked list is 11->11->11->21->43->43->60 then removeDuplicates() should convert the list to 11->21->43->60.

Algorithm: Traverse the list from the head (or start) node. While traversing, compare each node with its next node. If the data of the next node is the same as the current node then delete the next node. Before we delete a node, we need to store the next pointer of the node

Implementation: Functions other than removeDuplicates() are just to create a linked list and test removeDuplicates().

C++ Java

```
// Java program to remove duplicates from a sorted linked list
class LinkedList
{
    Node head; // head of list

    /* Linked list Node*/
    class Node
    {
        int data;
        Node next;
        Node(int d) {data = d; next = null; }
    }

    void removeDuplicates()
    {
        /*Another reference to head*/
        Node curr = head;
```

```
            Node temp = curr;
            while(temp!=null && temp.data.equals(curr.data)) {
                temp = temp.next;
            }
            /*Set current node next to the next different
            element denoted by temp*/
            curr.next = temp;
            curr = curr.next;
        }
    }

    /* Utility functions */

    /* Inserts a new Node at front of the list. */
    public void push(int new_data)
    {
        /* 1 & 2: Allocate the Node &
                  Put in the data*/
        Node new_node = new Node(new_data);

        /* 3. Make next of new Node as head */
        new_node.next = head;

        /* 4. Move the head to point to new Node */
        head = new_node;
    }

    /* Function to print linked list */
    void printList()
    {
        Node temp = head;
        while (temp != null)
        {
            System.out.print(temp.data+" ");
            temp = temp.next;
        }
        System.out.println();
    }
```

```
    public static void main(String args[])
    {
        LinkedList llist = new LinkedList();
        llist.push(13);
        llist.push(13);
        llist.push(11);
        llist.push(11);
        llist.push(11);

        System.out.println("List before removal of duplicates");
        llist.printList();

        llist.removeDuplicates();

        System.out.println("List after removal of elements");
        llist.printList();
    }
}
```

Output

```
Linked list before duplicate removal  11 11 11 13 13 20
Linked list after duplicate removal  11 13 20
```

Time Complexity: O(n) where n is the number of nodes in the given linked list.

Space Complexity: O(1) , as there is no extra space used.