Day 12

Assignment 5:

To efficiently remove duplicates from a sorted linked list, you can traverse the list while keeping track of the current and next nodes. Whenever you encounter consecutive nodes with the same value, you simply bypass the duplicate node by adjusting the next pointer of the current node. This way, you can remove duplicates in a single pass through the linked list without needing additional data structures.

A)

Here's the algorithm to remove duplicates from a sorted linked list:

Initialize Pointers: Initialize two pointers, current and next, to point to the head of the linked list.

Traverse the List: Iterate through the linked list until current reaches the last node.

Check for Duplicates:

If the value of current and next nodes are equal, bypass the next node by updating the next pointer of current to skip the duplicate node.

Otherwise, move both current and next pointers to the next nodes in the list.

Repeat: Continue this process until current reaches the last node.

Return the Modified Linked List: The modified linked list will have all duplicates removed.

Here's how you can implement this algorithm in Java:

```
package linkedlist;
public class ListNoderm {
        public static ListNode removeDuplicates(ListNode head) {
    if (head == null) return null;
    ListNode current = head;
    while (current != null && current.next != null) {
      if (current.val == current.next.val) {
        current.next = current.next.next;
      } else {
        current = current.next; // Move to the next node
      }
    }
    return head;
  }

  public static void printList(ListNode head) {
    ListNode current = head;
```

```
        while (current != null) {
            System.out.print(current.val + " ");
            current = current.next;
        }
        System.out.println();
    }

    public static void main(String[] args) {
        ListNode head = new ListNode(1);
        head.next = new ListNode(1);
        head.next.next = new ListNode(2);
        head.next.next.next = new ListNode(3);
        head.next.next.next.next = new ListNode(3);
        head.next.next.next.next.next = new ListNode(4);
        head.next.next.next.next.next.next = new ListNode(4);
        head.next.next.next.next.next.next.next = new ListNode(5);

        System.out.println("Original List:");
        printList(head);

        head = removeDuplicates(head);

        System.out.println("List after removing duplicates:");
        printList(head);
    }
}
```

Explanation:

The remove Duplicates function removes duplicates from the sorted linked list by iteratively traversing the list and bypassing duplicate nodes.

The print List function prints the elements of the linked list for demonstration purposes.

In the main method, a sorted linked list with repeated elements is created, and then duplicates are removed using the remove Duplicates function.

Output:

When you run the provided code, the output will be:

Original Linked List:

1 -> 1 -> 2 -> 3 -> 3 -> null

Linked List after removing duplicates:

1 -> 2 -> 3 -> null

Summary:

This algorithm efficiently removes duplicates from a sorted linked list by adjusting pointers without needing additional data structures. It performs the removal in a single pass through the list, making it an efficient solution for this task.