**Task 5: Removing Duplicates from a Sorted Linked List**

**A sorted linked list has been constructed with repeated elements. Describe an algorithm to remove all duplicates from the linked list efficiently.**

**Sol:**

**Steps to Remove Duplicates from a Sorted Linked List**

1. **Initialize Pointers**:

• Use a pointer to traverse the linked list, starting from the head.

2. **Traversal and Removal**: • For each node, check if its value is the same as the next node’s value.

* If they are the same, bypass the next node by pointing the current node’s next to the node after the next node.
* If they are not the same, move the pointer to the next node.
* Continue this process until you reach the end of the list.

**Program:**

class ListNode {

int val;

ListNode next;

ListNode(int x) {

val = x;

next = null;

}

}

public class Solution {

public void removeDuplicates(ListNode head) {

ListNode current = head;

while (current != null && current.next != null) {

if (current.val == current.next.val) {

current.next = current.next.next;

} else {

current = current.next;

}

}

}

public static void main(String[] args) {

// Example usage:

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(4);

head.next.next.next.next.next.next.next.next = new ListNode(5);

Solution solution = new Solution();

solution.removeDuplicates(head);

// Print the modified linked list

ListNode current = head;

while (current != null) {

System.out.print(current.val + " ");

current = current.next;

}

// Expected output: 1 2 3 4 5

}

}