Practical 8

Aim : Write a Program to find the Merge point of two linked lists(sorted)

1. Linked Lists: A linked list is a linear data structure where each element (node) contains a reference (or link) to the next node in the sequence. Each node typically contains data and a pointer to the next node.
2. Merge Point: When two linked lists share a common segment, the node at which they start to share the same sequence of nodes is called the merge point. This means that after this point, both linked lists will have the same nodes.

Example Linked Lists

Consider the following two linked lists:

- **List A**: 1 → 3 → 5 → 8
- **List B**: 2 → 4 → 8

Initialize Pointers:

- Set **l1** to the head of List A (node with value **1**).
- Set **l2** to the head of List B (node with value **2**).

Traverse the Lists:

- **First Iteration**:
    - Compare **l1.data** (1) and **l2.data** (2).

- Since **1 < 2**, move **l1** to the next node.

Second Iteration:

- Compare **l1.data** (3) and **l2.data** (2).
- Since **3 > 2**, move **l2** to the next node.

Third Iteration:

- Compare **l1.data** (3) and **l2.data** (4).
- Since **3 < 4**, move **l1** to the next node.

Fourth Iteration:

- Compare **l1.data** (5) and **l2.data** (4).
- Since **5 > 4**, move **l2** to the next node.

Fifth Iteration:

- Compare **l1.data** (5) and **l2.data** (8).
- Since **5 < 8**, move **l1** to the next node.

Sixth Iteration:

Compare `l1.data` (8) and `l2.data` (8).

Since they are equal, return `l1` (or `l2`), which is the merge point.

Check for Merge Point:

At this point, `l1` and `l2` are both pointing to the node with value `8`, which is the merge point. The algorithm returns this node.

In this example, both lists merge at the node with the value **8**.

Algorithm :

- Initialize Pointers: Set l1 to the head of the first linked list.
- Set l2 to the head of the second linked list.
-
- Traverse the Lists: While l1 is not equal to l2: If l1.data is less than l2.data: Move l1 to the next node (l1 = l1.next).
-
- If l1.data is greater than l2.data: Move l2 to the next node (l2 = l2.next).
-
- If l1.data is equal to l2.data: Return the node l1 (or l2, since they are equal).
-
-
-
- Check for Merge Point:

If l1 equals l2, return the node (this is the merge point).

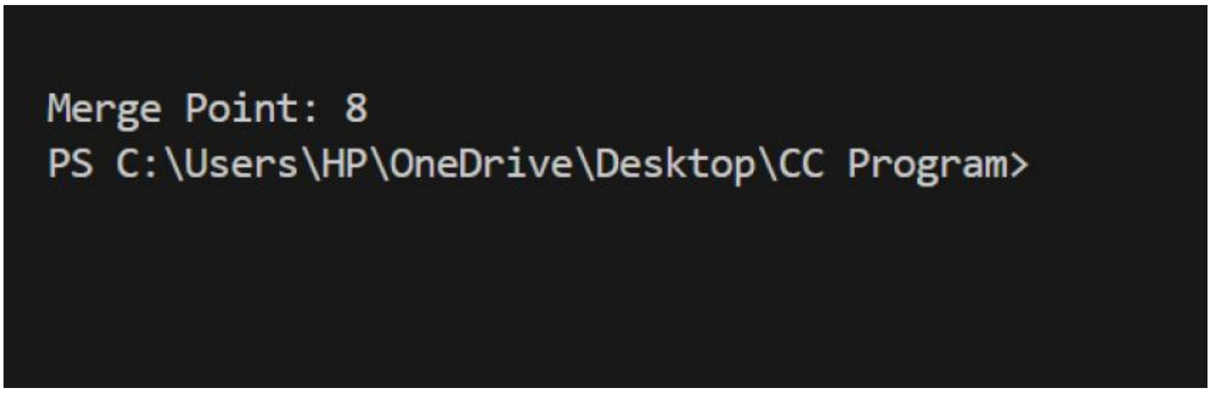
If both pointers reach the end (null), return null (indicating no merge point).

If a merge point is found, print the data of the merge point node.

If no merge point is found, print "No Merge Point found."

```
// Class representing a single node in a linked list
class Node {
int data; // Data stored in the node
Node next; // Reference to the next node in the list
// Constructor to initialize the node with data
Node(int data) {
this.data = data;
}
}
// Class to find the merge point of two linked lists
public class MergePointLinkedList {
// Method to find the merge point of two linked lists
public static Node findMergePoint(Node l1, Node l2) {
// Traverse both lists until the pointers meet
while (l1 != l2) {
// If the data in l1 is less than that in l2, move l1 to the next
node
if (l1.data < l2.data) {
l1 = l1.next;
}
// If the data in l1 is greater than that in l2, move l2 to the next node
else if (l1.data > l2.data) {
l2 = l2.next;
}
// If the data values are equal, return the current node as the merge point
else {
return l1;
}
}
// Return the node where the two lists intersect (or null if they don't)
return l1; // or return l2 (since they intersect at the same point)
}
// Main method to execute the program
public static void main(String[] args) {
// Create the intersecting node
Node intersectNode = new Node(8);
// Create the first linked list: 1 -> 3 -> 5 -> 8
Node l1 = new Node(1);
```

```
l1.next = new Node(3);
l1.next.next = new Node(5);
l1.next.next.next = intersectNode; // Link to the intersecting
node
// Create the second linked list: 2 -> 4 -> 8
Node l2 = new Node(2);
l2.next = new Node(4);
l2.next.next = intersectNode; // Link to the same intersecting
node
// Find the merge point of the two linked lists
Node mergePoint = findMergePoint(l1, l2);
// Print the result
if (mergePoint != null) {
System.out.println("Merge Point: " + mergePoint.data); //
Output the data of the merge point
} else {
System.out.println("No Merge Point found."); // Indicate no
merge point exists
}
}
}
```



```
Merge Point: 8
PS C:\Users\HP\OneDrive\Desktop\CC Program>
```