Assignment-5

To efficiently remove duplicates from a sorted linked list, you can traverse the list while maintaining two pointers: one pointing to the current node and another to the next distinct node.

Here's the algorithm:

1. **Initialize Pointers**: Initialize two pointers, **current** and **nextDistinct**, to point to the head of the linked list.
2. **Traverse the Linked List**:
   - Start traversing the list with the **current** pointer.
   - At each step, check if the data of the current node is equal to the data of the next node.
   - If they are equal, move the **nextDistinct** pointer to the next distinct node by skipping all consecutive nodes with the same data.
   - If they are not equal, update the **next** pointer of the last distinct node to point to the next distinct node (**nextDistinct**), effectively removing duplicates.
   - Move both **current** and **nextDistinct** pointers to their next nodes.
3. **Update Pointers**: Continue this process until the **current** pointer reaches the end of the linked list.
4. **Handle the End**: Finally, update the **next** pointer of the last distinct node to **null** to mark the end of the list.

This algorithm ensures that all duplicates are removed from the sorted linked list efficiently while maintaining the order of elements.