

**Name: Syed Muhammad Fahad Fiaz**

**Roll No: BSSEM-S24-036**

**Section: SE 3A**

**Subject: Data Structure and Algorithms**

**Submitted To: Sir Rasikh Ali**

**Submission Date:**

**Assignment 8**

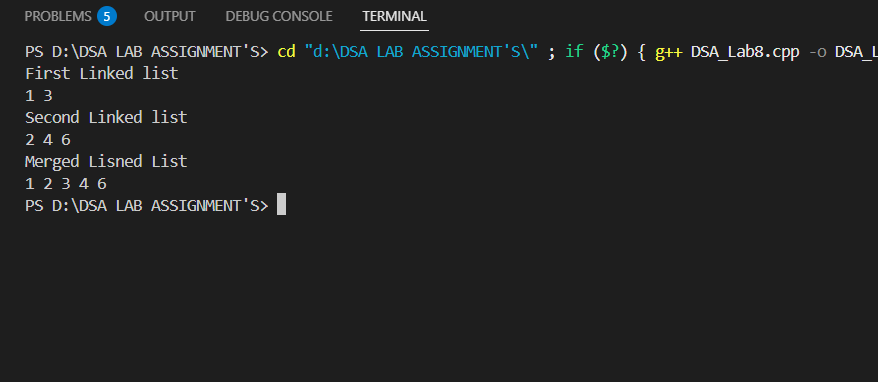
**DSA LAB TASK’S**

**LAB 8: Merge two LinkedList**

**Explanation:**

* **Node Class**:
* This defines a "node" (or element) in the linked list. Each node stores a value (data) and a pointer (next) that points to the next node in the list.
* **SinglyLinkedList Class**:
* This is a container for the linked list. It has:
  + A head pointer, which points to the first node of the list.
  + A method insert(), which adds new values to the end of the list.
  + A method print(), which prints the values of all nodes in the list.
* **mergeSinglyLinkedLists() Function**:
* This function takes two sorted linked lists and merges them into a single sorted list. It compares the first node of each list and recursively links them in order.
* It checks which node (from the two lists) has the smaller value and sets that node to come next in the merged list. This continues until all nodes are merged.

**Output**

****

**TASK 2: Merge two Doubly-Linked-List**

**Explanation:**

1. **DNode Class**:
   * Represents a **node** in the doubly linked list with three members:
     + val: stores the value of the node.
     + next: pointer to the next node in the list.
     + prev: pointer to the previous node in the list.
2. **DoublyLinkedList Class**:
   * Contains a pointer to the head of the list (head).
   * The insert() function adds a node with a given value at the end of the list.
   * The print() function prints all the nodes' values in the list.
3. **mergeDoublyLinkedLists Function**:
   * This is a **recursive** function that merges two sorted doubly linked lists into a single sorted doubly linked list.
   * It compares the first nodes of both lists and adds the smaller value to the merged list. The function recursively merges the remaining nodes.

**Process:**

1. **insert()**: Adds nodes to the doubly linked list.
   * If the list is empty, it sets head to the new node.
   * Otherwise, it traverses to the end of the list and appends the new node, updating the prev pointer of the new node to link it to the previous node.
2. **mergeDoublyLinkedLists()**:
   * If one list is empty, it returns the other list.
   * It compares the val of the first node from both lists, attaches the smaller node to the merged list, and recursively calls itself to merge the rest of the lists.
   * The prev pointer of each node is updated to maintain the doubly linked structure.
3. **print()**:
   * Prints the list from head to the last node by traversing through the next pointers.

**OUTPUT**

