**Lab Task 08**



**Superior University Gold Campus**

|  |  |
| --- | --- |
| **Submitted to** | ****Mr. Rasikh Ali**** |
| **Submitted by** | **Javaid Ali** |
| **Roll No** | **SU92-BSSEM-S24-029 (Section – 3A)** |
| **Subject** | **Data Structures and Algorithms (Lab)** |
| **Class** | **BS – Software Engineering** |

# **Lab 08: Merge two LinkedLists**

**1-Create 2 Singly LinkedLists and Merge them and display them.**

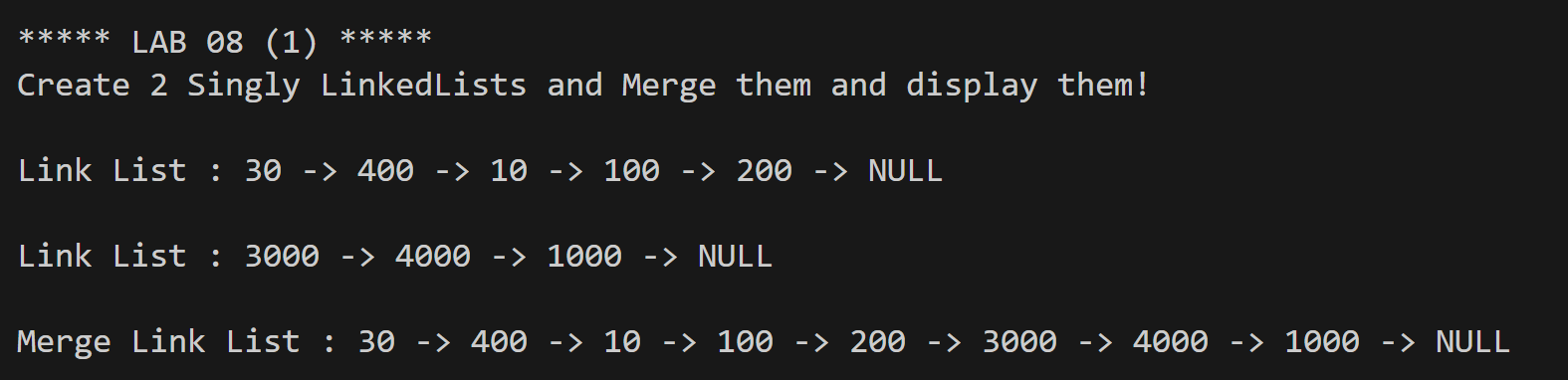
This program creates and merges two **singly linked lists**. The **Node** class stores data and a pointer to the next node, while the **LinkList** class manages the list operations.

Functions like **insert\_at\_start**, **insert\_at\_end**, and **insert\_at\_pos** allow inserting nodes at the beginning, end, and specific positions, respectively.

The **merge\_list** function combines two lists by appending the second list to the end of the first list. The program demonstrates this by creating two lists, inserting nodes, merging them, and displaying the merged list.

The **~LinkList** destructor ensures proper memory cleanup by deleting all nodes when the list is destroyed. This showcases efficient list manipulation and merging in a singly linked list.

**Outputs:**



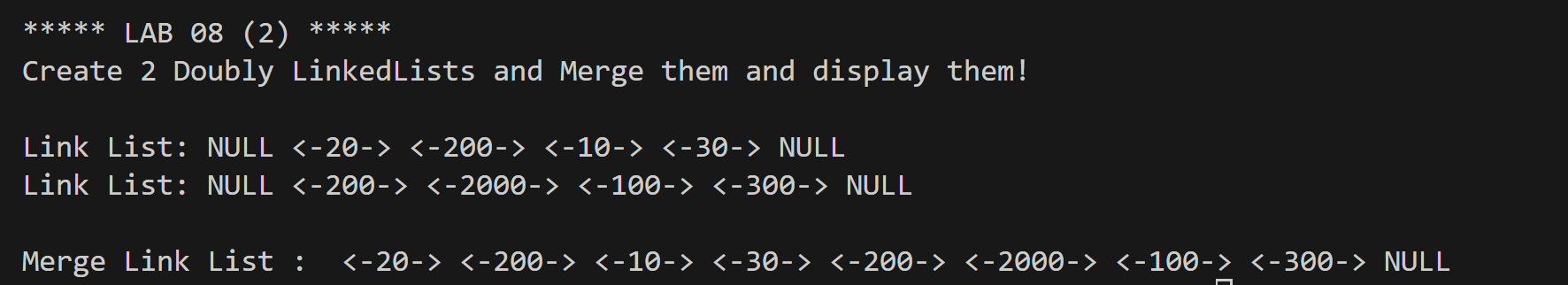
**2-Create 2 Doubly LinkedLists and Merge them and display them.**

This program demonstrates the creation and merging of two **doubly linked lists**. The **Node** class contains data, a pointer to the next node (next), and a pointer to the previous node (prev). The **LinkList** class provides functions to insert nodes at the **start**, **end**, and **specific positions** using **insert\_at\_start**, **insert\_at\_end**, and **insert\_at\_pos**.

The **merge\_list** function combines two lists by appending the second list to the end of the first list, ensuring the **next** and **prev** pointers are correctly updated to maintain the doubly linked structure. The program creates two lists, inserts nodes, merges them, and displays the merged list.

The **~LinkList** destructor ensures proper memory cleanup by deleting all nodes when the list is destroyed. This highlights efficient manipulation and merging of doubly linked lists.

**Outputs:**

****