

**Symbiosis Institute of Technology**

**Faculty of Engineering**

**CSE- Academic Year 2023-24**

**Data Structures – Lab Batch 2022-26**

| **Lab Assignment No:- 4,5,6** | |
| --- | --- |
|  | |
| **Name of Student** |  |
| **PRN No.** |  |
| **Batch** |  |
| **Class** |  |
| **Academic Year & Semester** |  |
| **Date of Submission** |  |
|  | |
| **Title of Assignment:** | 1. DS LAB Assignment\_4\_5\_6   SLL, CLL, DLL perform:   1. Create 2. display 3. Insert\_start 4. Insert\_last 5. Insert\_intermediate position 6. Insert\_after element 7. Delete\_Start 8. Delete\_end 9. Delete\_intermediate position 10. Delete\_after element 11. Reverse 12. Concat two linked list 13. Merge two linked list |
| **Theory:** | **Assignment-4**  **Write a menu-driven program that implements singly linked list for the following operations: create, insert, delete, reverse, concatenate**  A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers as shown in the below image:  **Linked List Operations: Traverse, Insert and Delete**   * Traversal - access each element of the linked list. * Insertion - adds a new element to the linked list. * Deletion - removes the existing elements. * Search - find a node in the linked list. * Sort - sort the nodes of the linked list  Traverse a Linked List Displaying the contents of a linked list is very simple. We keep moving the temp node to the next one and display its contents.  When temp is NULL, we know that we have reached the end of the linked list so we get out of the while loop. Insert Elements to a Linked List You can add elements to either the beginning, middle or end of the linked list. 1. Insert at the beginning  * Allocate memory for new node * Store data * Change next of new node to point to head * Change head to point to recently created node  Insert Elements to a Linked List You can add elements to either the beginning, middle or end of the linked list. 1. Insert at the beginning  * Allocate memory for new node * Store data * Change next of new node to point to head * Change head to point to recently created node  2. Insert at the End  * Allocate memory for new node * Store data * Traverse to last node * Change next of last node to recently created node  3. Insert at the Middle  * Allocate memory and store data for new node * Traverse to node just before the required position of new node * Change next pointers to include new node in between  Delete from a Linked List You can delete either from the beginning, end or from a particular position. 1. Delete from beginning  * Point head to the second node  2. Delete from end  * Traverse to second last element * Change its next pointer to null  3. Delete from middle  * Traverse to element before the element to be deleted * Change next pointers to exclude the node from the chain   **Search an Element on a Linked List**  You can search an element on a linked list using a loop using the following steps. We are finding item on a linked list.   * Make head as the current node. * Run a loop until the current node is NULL because the last element points to NULL. * In each iteration, check if the key of the node is equal to item. If it the key matches the item, return true otherwise return false.   **Assignment-5**  **Write a menu driven program that implements circular linked list for the following operations:Create, insert, delete, reverse, concatenate.** Singly Linked List as Circular In singly linked list, the next pointer of the last node points to the first node. Basic Operations Following are the important operations supported by a circular list.   * **insert** − Inserts an element at the start of the list. * **delete** − Deletes an element from the start of the list. * **display** − Displays the list.   **Assignment-6**  **Write a menu driven program that implements doubly linked list for the following operations:Create, insert, delete, reverse, concatenate.**  A **D**oubly **L**inked **L**ist (DLL) contains an extra pointer, typically called *previous pointer*, together with next pointer and data which are there in singly linked list.    **1) Add a node at the front:**  **2) Add a node after a given node.:**  **3) Add a node at the end:**  **4) Add a node before a given node:** |
| **Source Code/Algorithm/Flow Chart:** |  |
| **Output Screenshots (if applicable)** |  |
| **Conclusion** | Thus we have studied different Linked list algorithms and their operations |