```
/* PRACTICAL-5: Write a menu driven program to implement following operations on the singly linked list.
(a) Insert a node at the front of the linked list
(b) Insert a node at the end of the linked list
(c) Insert a node such that linked list is in ascending order
(d) Delete a First node of the linked list
(e) Delete a node before specified position
(f) Delete a node after specified position.

Name: Angat Shah
Enrollment No: 202203103510097
Branch: B.Tech Computer Science and Engineering */

// CODE:

import java.util.Scanner;
class Node {
    int data;
    Node next;
    Node(int data) {
        this.data = data;
        this.next = null;
    }
}
class LinkedList {
    Node head;
    LinkedList() {
        head = null;
    }
    // Insert a node at the front of the linked list
    void insertAtFront(int data) {
        Node newNode = new Node(data);
        newNode.next = head;
        head = newNode;
    }
    // Insert a node at the end of the linked list
    void insertAtEnd(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
            return;
        }
        Node temp = head;
        while (temp.next != null) {
            temp = temp.next;
        }
        temp.next = newNode;
    }
    // Insert a node in ascending order
    void insertInAscendingOrder(int data) {
        Node newNode = new Node(data);
        if (head == null || head.data >= data) {
            newNode.next = head;
            head = newNode;
            return;
        }
        Node current = head;
        while (current.next != null && current.next.data < data) {
            current = current.next;
```

```
        }
        newNode.next = current.next;
        current.next = newNode;
    }
    // Delete the first node of the linked list
    void deleteFirstNode() {
        if (head == null) {
            System.out.println("-->> List is empty");
            return;
        }
        head = head.next;
    }
    // Delete a node before specified position
    void deleteNodeBeforePosition(int position) {
        if (head == null || position < 1) {
            System.out.println("Invalid position or list is empty");
            return;
        }
        Node temp = head;
        for (int i = 1; temp != null && i < position - 1; i++) {
            temp = temp.next;
        }
        if (temp == null || temp.next == null) {
            System.out.println("Position out of range");
            return;
        }
        temp.next = temp.next.next;
    }
    // Delete a node after specified position
    void deleteNodeAfterPosition(int position) {
        if (head == null) {
            System.out.println("List is empty");
            return;
        }
        Node temp = head;
        for (int i = 1; temp != null && i < position; i++) {
            temp = temp.next;
        }
        if (temp == null || temp.next == null) {
            System.out.println("Position out of range");
            return;
        }
        temp.next = temp.next.next;
    }
    // Display the linked list
    void display() {
        Node temp = head;
        while (temp != null) {
            System.out.print(temp.data + " " + temp.next + " ");
            temp = temp.next;
        }
        System.out.println();
    }
}
public class practical5 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        LinkedList linkedList = new LinkedList();
```

```
        int choice;
        do {
            System.out.println("\n--> Operation Menu:");
            System.out.println("1. Insert at the front");
            System.out.println("2. Insert at the end");
            System.out.println("3. Insert in ascending order");
            System.out.println("4. Delete first node");
            System.out.println("5. Delete node before specified position");
            System.out.println("6. Delete node after specified position");
            System.out.println("7. Display linked list");
            System.out.println("0. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.next().charAt(0);
            switch (choice) {
                case '1':
                    System.out.print("-->> Enter Data to Insert at the Front: ");
                    int frontData = scanner.nextInt();
                    linkedList.insertAtFront(frontData);
                    break;
                case '2':
                    System.out.print("-->> Enter Data to Insert at the End: ");
                    int endData = scanner.nextInt();
                    linkedList.insertAtEnd(endData);
                    break;
                case '3':
                    System.out.print("-->> Enter Data to Insert in Ascending Order: ");
                    int ascData = scanner.nextInt();
                    linkedList.insertInAscendingOrder(ascData);
                    break;
                case '4':
                    linkedList.deleteFirstNode();
                    break;
                case '5':
                    System.out.print("-->> Enter position before which to delete: ");
                    int posBefore = scanner.nextInt();
                    linkedList.deleteNodeBeforePosition(posBefore);
                    break;
                case '6':
                    System.out.print("-->> Enter position after which to delete: ");
                    int posAfter = scanner.nextInt();
                    linkedList.deleteNodeAfterPosition(posAfter);
                    break;
                case '7':
                    linkedList.display();
                    break;
                case '0':
                    System.out.println("#Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice");
            }
        } while (choice != '0');
        scanner.close();
    }
}
```