

1 /* PRACTICAL-5: Write a menu driven program to implement following operations on the singly linked list.

2 (a) Insert a node at the front of the linked list

3 (b) Insert a node at the end of the linked list

4 (c) Insert a node such that linked list is in ascending order

5 (d) Delete a First node of the linked list

6 (e) Delete a node before specified position

7 (f) Delete a node after specified position.

8

9 Name: Angat Shah

10 Enrollment No: 202203103510097

11 Branch: B.Tech Computer Science and Engineering */

12

13 // CODE:

14

15 import java.util.Scanner;

16 class Node {

17 int data;

18 Node next;

19 Node(int data) {

20 this.data = data;

21 this.next = null;

22 }

23 }

24 class LinkedList {

25 Node head;

26 LinkedList() {

27 head = null;

28 }

29 // Insert a node at the front of the linked list

30 void insertAtFront(int data) {

31 Node newNode = new Node(data);

32 newNode.next = head;

33 head = newNode;

34 }

35 // Insert a node at the end of the linked list

36 void insertAtEnd(int data) {

37 Node newNode = new Node(data);

38 if (head == null) {

39 head = newNode;

40 return;

41 }

42 Node temp = head;

43 while (temp.next != null) {

44 temp = temp.next;

45 }

46 temp.next = newNode;

47 }

48 // Insert a node in ascending order

49 void insertInAscendingOrder(int data) {

50 Node newNode = new Node(data);

51 if (head == null || head.data >= data) {

52 newNode.next = head;

53 head = newNode;

54 return;

55 }

56 Node current = head;

57 while (current.next != null && current.next.data < data) {

58 current = current.next;

```

59     }
60     newNode.next = current.next;
61     current.next = newNode;
62 }
63 // Delete the first node of the linked list
64 void deleteFirstNode() {
65     if (head == null) {
66         System.out.println("-->> List is empty");
67         return;
68     }
69     head = head.next;
70 }
71 // Delete a node before specified position
72 void deleteNodeBeforePosition(int position) {
73     if (head == null || position < 1) {
74         System.out.println("Invalid position or list is empty");
75         return;
76     }
77     Node temp = head;
78     for (int i = 1; temp != null && i < position - 1; i++) {
79         temp = temp.next;
80     }
81     if (temp == null || temp.next == null) {
82         System.out.println("Position out of range");
83         return;
84     }
85     temp.next = temp.next.next;
86 }
87 // Delete a node after specified position
88 void deleteNodeAfterPosition(int position) {
89     if (head == null) {
90         System.out.println("List is empty");
91         return;
92     }
93     Node temp = head;
94     for (int i = 1; temp != null && i < position; i++) {
95         temp = temp.next;
96     }
97     if (temp == null || temp.next == null) {
98         System.out.println("Position out of range");
99         return;
100    }
101    temp.next = temp.next.next;
102 }
103 // Display the linked list
104 void display() {
105     Node temp = head;
106     while (temp != null) {
107         System.out.print(temp.data + " " + temp.next + " ");
108         temp = temp.next;
109     }
110     System.out.println();
111 }
112 }
113 public class practical5 {
114     public static void main(String[] args) {
115         Scanner scanner = new Scanner(System.in);
116         LinkedList linkedList = new LinkedList();

```

```

117 int choice;
118 do {
119     System.out.println("\n--> Operation Menu:");
120     System.out.println("1. Insert at the front");
121     System.out.println("2. Insert at the end");
122     System.out.println("3. Insert in ascending order");
123     System.out.println("4. Delete first node");
124     System.out.println("5. Delete node before specified position");
125     System.out.println("6. Delete node after specified position");
126     System.out.println("7. Display linked list");
127     System.out.println("0. Exit");
128     System.out.print("Enter your choice: ");
129     choice = scanner.nextInt().charAt(0);
130     switch (choice) {
131         case '1':
132             System.out.print("-->> Enter Data to Insert at the Front: ");
133             int frontData = scanner.nextInt();
134             linkedList.insertAtFront(frontData);
135             break;
136         case '2':
137             System.out.print("-->> Enter Data to Insert at the End: ");
138             int endData = scanner.nextInt();
139             linkedList.insertAtEnd(endData);
140             break;
141         case '3':
142             System.out.print("-->> Enter Data to Insert in Ascending Order: ");
143             int ascData = scanner.nextInt();
144             linkedList.insertInAscendingOrder(ascData);
145             break;
146         case '4':
147             linkedList.deleteFirstNode();
148             break;
149         case '5':
150             System.out.print("-->> Enter position before which to delete: ");
151             int posBefore = scanner.nextInt();
152             linkedList.deleteNodeBeforePosition(posBefore);
153             break;
154         case '6':
155             System.out.print("-->> Enter position after which to delete: ");
156             int posAfter = scanner.nextInt();
157             linkedList.deleteNodeAfterPosition(posAfter);
158             break;
159         case '7':
160             linkedList.display();
161             break;
162         case '0':
163             System.out.println("#Exiting...");
164             break;
165         default:
166             System.out.println("Invalid choice");
167     }
168 } while (choice != '0');
169 scanner.close();
170 }
171 }

```