

**ASSIGNMENT-01**

**Title: Doubly Linked List.**

**Name: Azizul Abedin Azmi**

**ID: 2022-1-60-130**

**Section: 03**

**Course Code: CSE207**

**Course Title: (Data Structures)**

**Date: 04/03/2024**

**Course Instructor:**

**Dr. Anup Kumar Paul**

**Associate Professor**

**Department of Computer Science and Engineering**

**Source Code:**

**Main.java:**

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        DoublyLinkedList list = new DoublyLinkedList();

        System.out.println("Enter the number of elements:");

        int n = scanner.nextInt();

        System.out.println("Enter the elements:");

        for (int i = 0; i < n; i++) {

            int data = scanner.nextInt();

            list.insertLast(data);

        }

        System.out.println("Original list:");

        list.display();

        System.out.println("Enter data to insert at first:");

        int firstData = scanner.nextInt();

        list.insertFirst(firstData);

        System.out.println("List after inserting at first:");

        list.display();

        System.out.println("Enter data to insert at last:");

        int lastData = scanner.nextInt();

        list.insertLast(lastData);

        System.out.println("List after inserting at last:");

        list.display();

        System.out.println("Enter data to insert at middle:");

        int middleData = scanner.nextInt();

        System.out.println("Enter position to insert at middle:");

        int position = scanner.nextInt();

        list.insertMiddle(middleData, position);

        System.out.println("List after inserting at middle:");

        list.display();

        System.out.println("Deleting first element:");

        list.deleteFirst();

        list.display();

        System.out.println("Deleting last element:");

        list.deleteLast();

        list.display();

        System.out.println("Enter position to delete from middle:");

        int deletePosition = scanner.nextInt();

        list.deleteMiddle(deletePosition);

        System.out.println("List after deleting from middle:");

        list.display();

        scanner.close();

    }

}

**DoublyLinkedList.java:**

class DoublyLinkedList {

    private Node head;

    private Node tail;

    DoublyLinkedList() {

        head = null;

        tail = null;

    }

    void insertFirst(int data) {

        Node newNode = new Node(data);

        if (head == null) {

            head = newNode;

            tail = newNode;

        } else {

            newNode.next = head;

            head.prev = newNode;

            head = newNode;

        }

    }

    void insertLast(int data) {

        Node newNode = new Node(data);

        if (tail == null) {

            head = newNode;

            tail = newNode;

        } else {

            tail.next = newNode;

            newNode.prev = tail;

            tail = newNode;

        }

    }

    void insertMiddle(int data, int position) {

        if (position <= 0) {

            System.out.println("Invalid position");

            return;

        }

        if (position == 1) {

            insertFirst(data);

            return;

        }

        Node newNode = new Node(data);

        Node current = head;

        for (int i = 1; i < position - 1 && current != null; i++) {

            current = current.next;

        }

        if (current == null) {

            System.out.println("Invalid position");

            return;

        }

        newNode.next = current.next;

        newNode.prev = current;

        if (current.next != null) {

            current.next.prev = newNode;

        }

        current.next = newNode;

    }

    void deleteFirst() {

        if (head == null) {

            System.out.println("List is empty");

            return;

        }

        head = head.next;

        if (head != null) {

            head.prev = null;

        } else {

            tail = null;

        }

    }

    void deleteLast() {

        if (tail == null) {

            System.out.println("List is empty");

            return;

        }

        tail = tail.prev;

        if (tail != null) {

            tail.next = null;

        } else {

            head = null;

        }

    }

    void deleteMiddle(int position) {

        if (position <= 0 || head == null) {

            System.out.println("Invalid position or list is empty");

            return;

        }

        if (position == 1) {

            deleteFirst();

            return;

        }

        Node current = head;

        for (int i = 1; i < position && current != null; i++) {

            current = current.next;

        }

        if (current == null) {

            System.out.println("Invalid position");

            return;

        }

        if (current == tail) {

            deleteLast();

            return;

        }

        current.prev.next = current.next;

        current.next.prev = current.prev;

    }

    void display() {

        Node current = head;

        while (current != null) {

            System.out.print(current.data + " ");

            current = current.next;

        }

        System.out.println();

    }

}

**Node.java:**

class Node {

    int data;

    Node prev;

    Node next;

    Node(int data) {

        this.data = data;

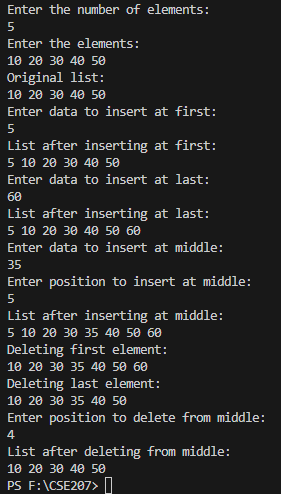
        this.prev = null;

        this.next = null;

    }

}

**Output:**

****