1. Singly Linked List

import java.util.\*;

class Node{

    int data;

    Node next;

    Node(int data){

        this.data=data;

        this.next=null;

    }

}

class SLL{

    public Node head;

    void insert(int data){

        if(head==null){

            head= new Node(data);

            return;

        }

        Node curr=head;

        while(curr.next != null ) curr=curr.next;

        curr.next=new Node(data);

    }

    void deletefrombeg(){

         if(head == null) {

        System.out.println("List is empty. Nothing to delete.");

        return;

        }

        Node curr=head.next;

        head=curr;

    }

    void deletefromend(){

        if(head == null) {

        System.out.println("List is empty. Nothing to delete.");

        return;

        }

        if(head.next == null) {

            head = null;

            return;

        }

        Node curr=head;

        Node prev=head;

        while(curr.next != null){

            prev=curr;

            curr=curr.next;

        }

        prev.next=null;

    }

    void deletefromiddle(int data){

        if(head == null) {

        System.out.println("List is empty. Nothing to delete.");

        return;

        }

        if(head.data == data) {

        head = head.next;

        return;

    }

        Node curr=head;

        Node prev=head;

        while(curr.data != data && curr != null){

            prev=curr;

            curr=curr.next;

        }

        if(curr == null) {

        System.out.println("Element not found.");

        return;

        }

        prev.next=curr.next;

    }

    void print(){

        if(head == null) {

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

        return;

        }

        Node curr=head;

        while(curr.next!= null){

            System.out.print(curr.data+"->");

            curr=curr.next;

        }

        System.out.println(curr.data);

    }

    }

public class SinglyLL{

    public static void main(String args[]){

        Scanner sc=new Scanner(System.in);

        SLL head= new SLL();

        int n=sc.nextInt();

        for(int i=0;i<n;i++){

           head.insert(sc.nextInt());

        }

        head.print();

        System.out.println("\nDelete from Begining");

        head.deletefrombeg();

        head.print();

        System.out.println("\nDelete from End");

        head.deletefromend();

        head.print();

        System.out.println("\nDelete from Middle");

        System.out.println("Enter the element to be deleted");

        int ele=sc.nextInt();

        head.deletefromiddle(ele);

        head.print();

    }

}

1. Doubly Linked List

import java.util.\*;

class Node{

    int data;

    Node next;

    Node prev;

    Node(int data){

        this.data=data;

        this.next=null;

        this.prev=null;

    }

}

class DLL{

    public Node head;

    void insert(int data){

        if(head==null){

            head= new Node(data);

            return;

        }

        Node curr=head;

        while(curr.next != null ) curr=curr.next;

        curr.next=new Node(data);

        curr.next.prev=curr;

    }

    void deletefrombeg(){

         if(head == null) {

        System.out.println("List is empty. Nothing to delete.");

        return;

        }

        Node curr=head.next;

        curr.prev=null;

        head=curr;

    }

    void deletefromend(){

        if(head == null) {

        System.out.println("List is empty. Nothing to delete.");

        return;

        }

        if(head.next == null) {

            head = null;

            return;

        }

        Node curr=head;

        Node prev=head;

        while(curr.next != null){

            prev=curr;

            curr=curr.next;

        }

        prev.next=null;

        curr.prev=null;

    }

    void deletefromiddle(int data){

        if(head == null) {

        System.out.println("List is empty. Nothing to delete.");

        return;

        }

        if(head.data == data) {

        head = head.next;

        return;

    }

        Node curr=head;

        Node prev=head;

        while(curr.data != data && curr.next != null){

            prev=curr;

            curr=curr.next;

        }

        if(curr.data != data) {

        System.out.println("Element not found.");

        return;

        }

        prev.next=curr.next;

        curr.next.prev=prev;

    }

    void leftToRightTraversal(){

        if(head == null) {

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

        return;

        }

        Node curr=head;

        while(curr.next!= null){

            System.out.print(curr.data+"->");

            curr=curr.next;

        }

        System.out.println(curr.data);

    }

    void rightToLeftTraversal(){

        if(head == null) {

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

        return;

        }

        Node curr=head;

        while(curr.next!= null){

            curr=curr.next;

        }

        while(curr.prev != null){

            System.out.print(curr.data+"->");

            curr=curr.prev;

        }

        System.out.println(curr.data);

    }

    }

public class DoublyLinkedList{

    public static void main(String args[]){

        Scanner sc=new Scanner(System.in);

        DLL head= new DLL();

        int n=sc.nextInt();

        for(int i=0;i<n;i++){

           head.insert(sc.nextInt());

        }

        head.leftToRightTraversal();

        System.err.println("Reverse the list");

        head.rightToLeftTraversal();

        System.out.println("\nDelete from Begining");

        head.deletefrombeg();

        head.leftToRightTraversal();

        System.out.println("\nDelete from End");

        head.deletefromend();

        head.leftToRightTraversal();

        System.out.println("\nDelete from Middle");

        System.out.println("Enter the element to be deleted");

        int ele=sc.nextInt();

        head.deletefromiddle(ele);

        head.leftToRightTraversal();

    }

}