**// LINEAR DATA STRUCTURE ASSIGNEMENT //**

**1. Write a program in JAVA to create and display Singly Linked List.**

**Test Data :**

**Input the number of nodes : 3**

**Input data for node 1 : 5**

**Input data for node 2 : 6**

**Input data for node 3 : 7**

**Expected Output :**

**Data entered in the list :**

**Data = 5**

**Data = 6**

**Data = 7**

import java.util.Scanner;  
  
class Node10 {  
 int data;  
 Node10 next;  
  
 Node10(int data) {  
 this.data = data;  
 this.next = null;  
 }  
}  
  
class SinglyLinkedList {  
 Node10 head;  
  
 void insert(int data) {  
 Node10 newNode = new Node10(data);  
 if (head == null) {  
 head = newNode;  
 return;  
 }  
 Node10 temp = head;  
 while (temp.next != null) {  
 temp = temp.next;  
 }  
 temp.next = newNode;  
 }  
  
 void display() {  
 Node10 temp = head;  
 while (temp != null) {  
 System.*out*.println("Data = " + temp.data);  
 temp = temp.next;  
 }  
 }  
}  
  
public class Main10 {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 SinglyLinkedList list = new SinglyLinkedList();  
 System.*out*.print("Input the number of nodes : ");  
 int n = sc.nextInt();  
 for (int i = 1; i <= n; i++) {  
 System.*out*.print("Input data for node " + i + " : ");  
 int data = sc.nextInt();  
 list.insert(data);  
 }  
 System.*out*.println("Data entered in the list :");  
 list.display();  
 sc.close();  
 }  
}

**// OUTPUT ->**

**Input the number of nodes : 3**

**Input data for node 1 : 5**

**Input data for node 2 : 6**

**Input data for node 3 : 7**

**Data entered in the list :**

**Data = 5**

**Data = 6**

**Data = 7**

**2. Write a program in JAVA to create a singly linked list of n nodes and**

**display it in reverse order.**

**Test Data :**

**Input the number of nodes : 3**

**Input data for node 1 : 5**

**Input data for node 2 : 6**

**Input data for node 3 : 7**

**Expected Output :**

**Data entered in the list are :**

**Data = 5**

**Data = 6**

**Data = 7**

**The list in reverse are :**

**Data = 7**

**Data = 6**

**Data = 5**

import java.util.Scanner;  
  
class Node11 {  
 int data;  
 Node11 next;  
  
 Node11(int data) {  
 this.data = data;  
 this.next = null;  
 }  
}  
  
class SinglyLinkedList\_1 {  
 Node11 head;  
  
 void insert(int data) {  
 Node11 newNode = new Node11(data);  
 if (head == null) {  
 head = newNode;  
 return;  
 }  
 Node11 temp = head;  
 while (temp.next != null) {  
 temp = temp.next;  
 }  
 temp.next = newNode;  
 }  
  
 void display() {  
 Node11 temp = head;  
 while (temp != null) {  
 System.*out*.println("Data = " + temp.data);  
 temp = temp.next;  
 }  
 }  
  
 void displayReverse(Node11 node) {  
 if (node == null) return;  
 displayReverse(node.next);  
 System.*out*.println("Data = " + node.data);  
 }  
  
 void displayInReverse() {  
 displayReverse(head);  
 }  
}  
  
public class Main11 {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 SinglyLinkedList\_1 list = new SinglyLinkedList\_1();  
 System.*out*.print("Input the number of nodes : ");  
 int n = sc.nextInt();  
 for (int i = 1; i <= n; i++) {  
 System.*out*.print("Input data for node " + i + " : ");  
 int data = sc.nextInt();  
 list.insert(data);  
 }  
 System.*out*.println("Data entered in the list :");  
 list.display();  
 System.*out*.println("The list in reverse are :");  
 list.displayInReverse();  
 sc.close();  
 }  
}

**OUTPUT ->**

**Input the number of nodes : 3**

**Input data for node 1 : 5**

**Input data for node 2 : 6**

**Input data for node 3 : 7**

**Data entered in the list :**

**Data = 5**

**Data = 6**

**Data = 7**

**The list in reverse are :**

**Data = 7**

**Data = 6**

**Data = 5**

**3. Write a program in JAVA to create a singly linked list of n nodes and**

**count the number of nodes.**

**Test Data :**

**Input the number of nodes : 3**

**Input data for node 1 : 5**

**Input data for node 2 : 6**

**Input data for node 3 : 7**

**Expected Output :**

**Data entered in the list are :**

**Data = 5**

**Data = 6**

**Data = 7**

**Total number of nodes = 3**

import java.util.Scanner;  
  
class Node\_12 {  
 int data;  
 Node\_12 next;  
  
 Node\_12(int data) {  
 this.data = data;  
 this.next = null;  
 }  
}  
  
class SinglyLinked\_12 {  
 Node\_12 head;  
  
 void insert(int data) {  
 Node\_12 newNode = new Node\_12(data);  
 if (head == null) {  
 head = newNode;  
 return;  
 }  
 Node\_12 temp = head;  
 while (temp.next != null) {  
 temp = temp.next;  
 }  
 temp.next = newNode;  
 }  
  
 void display() {  
 Node\_12 temp = head;  
 while (temp != null) {  
 System.*out*.println("Data = " + temp.data);  
 temp = temp.next;  
 }  
 }  
  
 int countNodes() {  
 int count = 0;  
 Node\_12 temp = head;  
 while (temp != null) {  
 count++;  
 temp = temp.next;  
 }  
 return count;  
 }  
}  
  
public class Main\_12 {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 SinglyLinked\_12 list = new SinglyLinked\_12();  
 System.*out*.print("Input the number of nodes : ");  
 int n = sc.nextInt();  
 for (int i = 1; i <= n; i++) {  
 System.*out*.print("Input data for node " + i + " : ");  
 int data = sc.nextInt();  
 list.insert(data);  
 }  
 System.*out*.println("Data entered in the list :");  
 list.display();  
 System.*out*.println("Total number of nodes = " + list.countNodes());  
 sc.close();  
 }  
}

**OUTPUT ->**

**C:\Users\yadav\.jdks\openjdk-21.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.3.5\lib\idea\_rt.jar=54726:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.3.5\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath "C:\Users\yadav\OneDrive\Desktop\PRACTISE OF JAVA QUESTION\out\production\PRACTISE OF JAVA QUESTION" Main\_12**

**Input the number of nodes : 3**

**Input data for node 1 : 5**

**Input data for node 2 : 6**

**Input data for node 3 : 7**

**Data entered in the list :**

**Data = 5**

**Data = 6**

**Data = 7**

**Total number of nodes = 3**

**4. Write a program in JAVA to insert a new node at the beginning of a**

**Singly Linked List.**

**Test Data and Expected Output :**

**Input the number of nodes : 3**

**Input data for node 1 : 5**

**Input data for node 2 : 6**

**Input data for node 3 : 7**

**Data entered in the list are :**

**Data = 5**

**Data = 6**

**Data = 7**

**Input data to insert at the beginning of the list : 4**

**Data after inserted in the list are :**

**Data = 4**

**Data = 5**

**Data = 6**

**Data = 7**

import java.util.Scanner;  
  
class Node13 {  
 int data;  
 Node13 next;  
  
 Node13(int data) {  
 this.data = data;  
 this.next = null;  
 }  
}  
  
class Linked13 {  
 Node13 head;  
  
 void insertAtBeginning(int data) {  
 Node13 newNode = new Node13(data);  
 newNode.next = head;  
 head = newNode;  
 }  
  
 void insertAtEnd(int data) {  
 Node13 newNode = new Node13(data);  
 if (head == null) {  
 head = newNode;  
 return;  
 }  
 Node13 temp = head;  
 while (temp.next != null) {  
 temp = temp.next;  
 }  
 temp.next = newNode;  
 }  
  
 void display() {  
 Node13 temp = head;  
 while (temp != null) {  
 System.*out*.println("Data = " + temp.data);  
 temp = temp.next;  
 }  
 }  
}  
  
public class SinglyLinkedList13 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 Linked13 list = new Linked13();  
  
 System.*out*.print("Input the number of nodes: ");  
 int n = scanner.nextInt();  
  
 for (int i = 1; i <= n; i++) {  
 System.*out*.print("Input data for node " + i + ": ");  
 int data = scanner.nextInt();  
 list.insertAtEnd(data);  
 }  
  
 System.*out*.println("Data entered in the list are:");  
 list.display();  
  
 System.*out*.print("Input data to insert at the beginning of the list: ");  
 int newData = scanner.nextInt();  
 list.insertAtBeginning(newData);  
  
 System.*out*.println("Data after inserted in the list are:");  
 list.display();  
 }  
}

**OUTPUT ->**

**Input the number of nodes: 3**

**Input data for node 1: 5**

**Input data for node 2: 6**

**Input data for node 3: 7**

**Data entered in the list are:**

**Data = 5**

**Data = 6**

**Data = 7**

**Input data to insert at the beginning of the list: 4**

**Data after inserted in the list are:**

**Data = 4**

**Data = 5**

**Data = 6**

**Data = 7**

**5. Write a program in JAVA to insert a new node at the end of a Singly**

**Linked List.**

import java.util.Scanner;  
  
class Node13 {  
 int data;  
 Node13 next;  
  
 Node13(int data) {  
 this.data = data;  
 this.next = null;  
 }  
}  
  
class Linked13 {  
 Node13 head;  
  
 void insertAtBeginning(int data) {  
 Node13 newNode = new Node13(data);  
 newNode.next = head;  
 head = newNode;  
 }  
  
 void insertAtEnd(int data) {  
 Node13 newNode = new Node13(data);  
 if (head == null) {  
 head = newNode;  
 return;  
 }  
 Node13 temp = head;  
 while (temp.next != null) {  
 temp = temp.next;  
 }  
 temp.next = newNode;  
 }  
  
 void display() {  
 Node13 temp = head;  
 while (temp != null) {  
 System.*out*.println("Data = " + temp.data);  
 temp = temp.next;  
 }  
 }  
}  
  
public class SinglyLinkedList13 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 Linked13 list = new Linked13();  
  
 System.*out*.print("Input the number of nodes: ");  
 int n = scanner.nextInt();  
  
 for (int i = 1; i <= n; i++) {  
 System.*out*.print("Input data for node " + i + ": ");  
 int data = scanner.nextInt();  
 list.insertAtEnd(data);  
 }  
  
 System.*out*.println("Data entered in the list are:");  
 list.display();  
  
// System.out.print("Input data to insert at the beginning of the list: ");  
// int newData = scanner.nextInt();  
// list.insertAtBeginning(newData);  
  
 System.*out*.print("Input data to insert at the end of the list: ");  
 int newData = scanner.nextInt();  
 list.insertAtEnd(newData);  
  
  
 System.*out*.println("Data after inserted in the list are:");  
 list.display();  
 }  
}

**OUTPUT ->**

**Input the number of nodes: 3**

**Input data for node 1: 5**

**Input data for node 2: 6**

**Input data for node 3: 7**

**Data entered in the list are:**

**Data = 5**

**Data = 6**

**Data = 7**

**Input data to insert at the end of the list: 8**

**Data after inserted in the list are:**

**Data = 5**

**Data = 6**

**Data = 7**

**Data = 8**

**6. Write a program in JAVA to insert a new node at the middle of Singly**

**Linked List.**

import java.util.Scanner;  
  
class Node13 {  
 int data;  
 Node13 next;  
  
 Node13(int data) {  
 this.data = data;  
 this.next = null;  
 }  
}  
  
class Linked13 {  
 Node13 head;  
  
 void insertAtBeginning(int data) {  
 Node13 newNode = new Node13(data);  
 newNode.next = head;  
 head = newNode;  
 }  
  
 void insertAtMiddle(int data) {  
 if (head == null) {  
 head = new Node13(data);  
 return;  
 }  
 Node13 slow = head, fast = head;  
 Node13 prev = null;  
 while (fast != null && fast.next != null) {  
 prev = slow;  
 slow = slow.next;  
 fast = fast.next.next;  
 }  
 Node13 newNode = new Node13(data);  
 if (prev != null) {  
 newNode.next = prev.next;  
 prev.next = newNode;  
 }  
 }  
  
 /\* void insertAtEnd(int data) {  
 Node13 newNode = new Node13(data);  
 if (head == null) {  
 head = newNode;  
 return;  
 }  
 Node13 temp = head;  
 while (temp.next != null) {  
 temp = temp.next;  
 }  
 temp.next = newNode;  
 } \*/  
  
 void display() {  
 Node13 temp = head;  
 while (temp != null) {  
 System.*out*.println("Data = " + temp.data);  
 temp = temp.next;  
 }  
 }  
}  
  
public class SinglyLinkedList13 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 Linked13 list = new Linked13();  
  
 System.*out*.print("Input the number of nodes: ");  
 int n = scanner.nextInt();  
  
 for (int i = 1; i <= n; i++) {  
 System.*out*.print("Input data for node " + i + ": ");  
 int data = scanner.nextInt();  
 // list.insertAtEnd(data); // Commented out insertAtEnd  
 list.insertAtBeginning(data);  
 }  
  
 System.*out*.println("\nData entered in the list are:");  
 list.display();  
  
 System.*out*.print("\nInput data to insert at the middle of the list: ");  
 int newData = scanner.nextInt();  
 list.insertAtMiddle(newData);  
  
 System.*out*.println("\nData after inserted in the list are:");  
 list.display();  
 }  
}

**OUTPUT ->**

**Input the number of nodes: 3**

**Input data for node 1: 10**

**Input data for node 2: 20**

**Input data for node 3: 30**

**Data entered in the list are:**

**Data = 30**

**Data = 20**

**Data = 10**

**Input data to insert at the middle of the list: 15**

**Data after inserted in the list are:**

**Data = 30**

**Data = 15**

**Data = 20**

**Data = 10**

**7. Write a program in C to delete first node of Singly Linked List.**

**Test Data :**

**7. Write a program in java to delete first node of Singly Linked List.**

import java.util.Scanner;  
  
class Node13 {  
 int data;  
 Node13 next;  
  
 Node13(int data) {  
 this.data = data;  
 this.next = null;  
 }  
}  
  
class Linked13 {  
 Node13 head;  
  
 void insertAtBeginning(int data) {  
 Node13 newNode = new Node13(data);  
 newNode.next = head;  
 head = newNode;  
 }  
  
 void insertAtMiddle(int data) {  
 if (head == null) {  
 head = new Node13(data);  
 return;  
 }  
 Node13 slow = head, fast = head;  
 Node13 prev = null;  
 while (fast != null && fast.next != null) {  
 prev = slow;  
 slow = slow.next;  
 fast = fast.next.next;  
 }  
 Node13 newNode = new Node13(data);  
 if (prev != null) {  
 newNode.next = prev.next;  
 prev.next = newNode;  
 }  
 }  
  
 void deleteFirstNode() {  
 if (head == null) {  
 System.*out*.println("List is already empty.");  
 return;  
 }  
 head = head.next;  
 }  
  
 void deleteMiddleNode() {  
 if (head == null || head.next == null) {  
 System.*out*.println("List is too small to delete the middle node.");  
 return;  
 }  
 Node13 slow = head, fast = head;  
 Node13 prev = null;  
 while (fast != null && fast.next != null) {  
 prev = slow;  
 slow = slow.next;  
 fast = fast.next.next;  
 }  
 if (prev != null) {  
 prev.next = slow.next;  
 }  
 }  
  
 void display() {  
 Node13 temp = head;  
 while (temp != null) {  
 System.*out*.println("Data = " + temp.data);  
 temp = temp.next;  
 }  
 }  
}  
  
public class SinglyLinkedList13 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 Linked13 list = new Linked13();  
  
 System.*out*.print("Input the number of nodes: ");  
 int n = scanner.nextInt();  
  
 for (int i = 1; i <= n; i++) {  
 System.*out*.print("Input data for node " + i + ": ");  
 int data = scanner.nextInt();  
 list.insertAtBeginning(data);  
 }  
  
 System.*out*.println("\nData entered in the list are:");  
 list.display();  
  
 System.*out*.println("\nDeleting the first node...");  
 list.deleteFirstNode();  
 System.*out*.println("\nData after first node deletion:");  
 list.display();  
  
// System.out.println("\nDeleting the middle node...");  
// list.deleteMiddleNode();  
// System.out.println("\nData after middle node deletion:");  
// list.display();  
 }  
}

**OUTPUT ->**

**Input the number of nodes: 4**

**Input data for node 1: 10**

**Input data for node 2: 20**

**Input data for node 3: 30**

**Input data for node 4: 40**

**Data entered in the list are:**

**Data = 40**

**Data = 30**

**Data = 20**

**Data = 10**

**Deleting the first node...**

**Data after first node deletion:**

**Data = 30**

**Data = 20**

**Data = 10**

**8. Write a program in JAVA to delete a node from the middle of Singly**

**Linked List.**

import java.util.Scanner;  
  
class Node13 {  
 int data;  
 Node13 next;  
  
 Node13(int data) {  
 this.data = data;  
 this.next = null;  
 }  
}  
  
class Linked13 {  
 Node13 head;  
  
 void insertAtBeginning(int data) {  
 Node13 newNode = new Node13(data);  
 newNode.next = head;  
 head = newNode;  
 }  
  
 void insertAtMiddle(int data) {  
 if (head == null) {  
 head = new Node13(data);  
 return;  
 }  
 Node13 slow = head, fast = head;  
 Node13 prev = null;  
 while (fast != null && fast.next != null) {  
 prev = slow;  
 slow = slow.next;  
 fast = fast.next.next;  
 }  
 Node13 newNode = new Node13(data);  
 if (prev != null) {  
 newNode.next = prev.next;  
 prev.next = newNode;  
 }  
 }  
  
 void deleteFirstNode() {  
 if (head == null) {  
 System.*out*.println("List is already empty.");  
 return;  
 }  
 head = head.next;  
 }  
  
 void deleteMiddleNode() {  
 if (head == null || head.next == null) {  
 System.*out*.println("List is too small to delete the middle node.");  
 return;  
 }  
 Node13 slow = head, fast = head;  
 Node13 prev = null;  
 while (fast != null && fast.next != null) {  
 prev = slow;  
 slow = slow.next;  
 fast = fast.next.next;  
 }  
 if (prev != null) {  
 prev.next = slow.next;  
 }  
 }  
  
 void display() {  
 Node13 temp = head;  
 while (temp != null) {  
 System.*out*.println("Data = " + temp.data);  
 temp = temp.next;  
 }  
 }  
}  
  
public class SinglyLinkedList13 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 Linked13 list = new Linked13();  
  
 System.*out*.print("Input the number of nodes: ");  
 int n = scanner.nextInt();  
  
 for (int i = 1; i <= n; i++) {  
 System.*out*.print("Input data for node " + i + ": ");  
 int data = scanner.nextInt();  
 list.insertAtBeginning(data);  
 }  
  
 System.*out*.println("\nData entered in the list are:");  
 list.display();  
  
// System.out.println("\nDeleting the first node...");  
// list.deleteFirstNode();  
// System.out.println("\nData after first node deletion:");  
// list.display();  
  
 System.*out*.println("\nDeleting the middle node...");  
 list.deleteMiddleNode();  
 System.*out*.println("\nData after middle node deletion:");  
 list.display();  
 }  
}

**OUTPUT ->**

**Input the number of nodes: 4**

**Input data for node 1: 15**

**Input data for node 2: 25**

**Input data for node 3: 35**

**Input data for node 4: 45**

**Data entered in the list are:**

**Data = 45**

**Data = 35**

**Data = 25**

**Data = 15**

**Deleting the middle node...**

**Data after middle node deletion:**

**Data = 45**

**Data = 35**

**Data = 15**

**9. Write a program in Java to delete the last node of Singly Linked**

**List.**

import java.util.Scanner;  
  
class Node13 {  
 int data;  
 Node13 next;  
  
 Node13(int data) {  
 this.data = data;  
 this.next = null;  
 }  
}  
  
class Linked13 {  
 Node13 head;  
  
 void insertAtBeginning(int data) {  
 Node13 newNode = new Node13(data);  
 newNode.next = head;  
 head = newNode;  
 }  
  
 void insertAtMiddle(int data) {  
 if (head == null) {  
 head = new Node13(data);  
 return;  
 }  
 Node13 slow = head, fast = head;  
 Node13 prev = null;  
 while (fast != null && fast.next != null) {  
 prev = slow;  
 slow = slow.next;  
 fast = fast.next.next;  
 }  
 Node13 newNode = new Node13(data);  
 if (prev != null) {  
 newNode.next = prev.next;  
 prev.next = newNode;  
 }  
 }  
  
 void deleteFirstNode() {  
 if (head == null) {  
 System.*out*.println("List is already empty.");  
 return;  
 }  
 head = head.next;  
 }  
  
 void deleteMiddleNode() {  
 if (head == null || head.next == null) {  
 System.*out*.println("List is too small to delete the middle node.");  
 return;  
 }  
 Node13 slow = head, fast = head;  
 Node13 prev = null;  
 while (fast != null && fast.next != null) {  
 prev = slow;  
 slow = slow.next;  
 fast = fast.next.next;  
 }  
 if (prev != null) {  
 prev.next = slow.next;  
 }  
 }  
  
 void deleteLastNode() {  
 if (head == null) {  
 System.*out*.println("List is already empty.");  
 return;  
 }  
 if (head.next == null) {  
 head = null;  
 return;  
 }  
 Node13 temp = head;  
 while (temp.next.next != null) {  
 temp = temp.next;  
 }  
 temp.next = null;  
 }  
  
 void display() {  
 Node13 temp = head;  
 while (temp != null) {  
 System.*out*.println("Data = " + temp.data);  
 temp = temp.next;  
 }  
 }  
}  
  
public class SinglyLinkedList13 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 Linked13 list = new Linked13();  
  
 System.*out*.print("Input the number of nodes: ");  
 int n = scanner.nextInt();  
  
 for (int i = 1; i <= n; i++) {  
 System.*out*.print("Input data for node " + i + ": ");  
 int data = scanner.nextInt();  
 list.insertAtBeginning(data);  
 }  
  
 System.*out*.println("\nData entered in the list are:");  
 list.display();  
  
// System.out.println("\nDeleting the first node...");  
// list.deleteFirstNode();  
// System.out.println("\nData after first node deletion:");  
// list.display();  
//  
// System.out.println("\nDeleting the middle node...");  
// list.deleteMiddleNode();  
// System.out.println("\nData after middle node deletion:");  
// list.display();  
  
 System.*out*.println("\nDeleting the last node...");  
 list.deleteLastNode();  
 System.*out*.println("\nData after last node deletion:");  
 list.display();  
 }  
}

**OUTPUT ->**

**Input the number of nodes: 5**

**Input data for node 1: 10**

**Input data for node 2: 20**

**Input data for node 3: 30**

**Input data for node 4: 40**

**Input data for node 5: 50**

**Data entered in the list are:**

**Data = 50**

**Data = 40**

**Data = 30**

**Data = 20**

**Data = 10**

**Deleting the last node...**

**Data after last node deletion:**

**Data = 50**

**Data = 40**

**Data = 30**

**Data = 20**

**10. Write a program in JAVA to search an existing element in a singly**

**linked list.**

import java.util.Scanner;  
  
class Node13 {  
 int data;  
 Node13 next;  
  
 Node13(int data) {  
 this.data = data;  
 this.next = null;  
 }  
}  
  
class Linked13 {  
 Node13 head;  
  
 void insertAtBeginning(int data) {  
 Node13 newNode = new Node13(data);  
 newNode.next = head;  
 head = newNode;  
 }  
  
 void insertAtMiddle(int data) {  
 if (head == null) {  
 head = new Node13(data);  
 return;  
 }  
 Node13 slow = head, fast = head;  
 Node13 prev = null;  
 while (fast != null && fast.next != null) {  
 prev = slow;  
 slow = slow.next;  
 fast = fast.next.next;  
 }  
 Node13 newNode = new Node13(data);  
 if (prev != null) {  
 newNode.next = prev.next;  
 prev.next = newNode;  
 }  
 }  
  
 void deleteFirstNode() {  
 if (head == null) {  
 System.*out*.println("List is already empty.");  
 return;  
 }  
 head = head.next;  
 }  
  
 void deleteMiddleNode() {  
 if (head == null || head.next == null) {  
 System.*out*.println("List is too small to delete the middle node.");  
 return;  
 }  
 Node13 slow = head, fast = head;  
 Node13 prev = null;  
 while (fast != null && fast.next != null) {  
 prev = slow;  
 slow = slow.next;  
 fast = fast.next.next;  
 }  
 if (prev != null) {  
 prev.next = slow.next;  
 }  
 }  
  
 void deleteLastNode() {  
 if (head == null) {  
 System.*out*.println("List is already empty.");  
 return;  
 }  
 if (head.next == null) {  
 head = null;  
 return;  
 }  
 Node13 temp = head;  
 while (temp.next.next != null) {  
 temp = temp.next;  
 }  
 temp.next = null;  
 }  
  
 boolean search(int key) {  
 Node13 temp = head;  
 while (temp != null) {  
 if (temp.data == key) {  
 return true;  
 }  
 temp = temp.next;  
 }  
 return false;  
 }  
  
 void display() {  
 Node13 temp = head;  
 while (temp != null) {  
 System.*out*.println("Data = " + temp.data);  
 temp = temp.next;  
 }  
 }  
}  
  
public class SinglyLinkedList13 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 Linked13 list = new Linked13();  
  
 System.*out*.print("Input the number of nodes: ");  
 int n = scanner.nextInt();  
  
 for (int i = 1; i <= n; i++) {  
 System.*out*.print("Input data for node " + i + ": ");  
 int data = scanner.nextInt();  
 list.insertAtBeginning(data);  
 }  
  
 System.*out*.println("\nData entered in the list are:");  
 list.display();  
  
// System.out.println("\nDeleting the first node...");  
// list.deleteFirstNode();  
// System.out.println("\nData after first node deletion:");  
// list.display();  
//  
// System.out.println("\nDeleting the middle node...");  
// list.deleteMiddleNode();  
// System.out.println("\nData after middle node deletion:");  
// list.display();  
//  
// System.out.println("\nDeleting the last node...");  
// list.deleteLastNode();  
// System.out.println("\nData after last node deletion:");  
// list.display();  
  
 System.*out*.print("\nEnter an element to search: ");  
 int searchKey = scanner.nextInt();  
 if (list.search(searchKey)) {  
 System.*out*.println("Element found in the list.");  
 } else {  
 System.*out*.println("Element not found in the list.");  
 }  
 }  
}

**OUTPUT ->**

**Input the number of nodes: 4**

**Input data for node 1: 10**

**Input data for node 2: 20**

**Input data for node 3: 30**

**Input data for node 4: 40**

**Data entered in the list are:**

**Data = 40**

**Data = 30**

**Data = 20**

**Data = 10**

**Enter an element to search: 10**

**Element found in the list.**

**11. Write a program in java to create and display a doubly linked list.**

import java.util.Scanner;  
  
class Node14 {  
 int data;  
 Node14 next, prev;  
  
 Node14(int data) {  
 this.data = data;  
 this.next = null;  
 this.prev = null;  
 }  
}  
  
class DoublyLinkedList14 {  
 Node14 head, tail;  
  
 void insertAtBeginning(int data) {  
 Node14 newNode = new Node14(data);  
 if (head == null) {  
 head = tail = newNode;  
 } else {  
 newNode.next = head;  
 head.prev = newNode;  
 head = newNode;  
 }  
 }  
  
 void insertAtEnd(int data) {  
 Node14 newNode = new Node14(data);  
 if (head == null) {  
 head = tail = newNode;  
 return;  
 }  
 tail.next = newNode;  
 newNode.prev = tail;  
 tail = newNode;  
 }  
  
 void insertAtMiddle(int data) {  
 if (head == null) {  
 head = tail = new Node14(data);  
 return;  
 }  
 Node14 slow = head, fast = head;  
 while (fast != null && fast.next != null) {  
 slow = slow.next;  
 fast = fast.next.next;  
 }  
 Node14 newNode = new Node14(data);  
 newNode.next = slow;  
 newNode.prev = slow.prev;  
 if (slow.prev != null) {  
 slow.prev.next = newNode;  
 } else {  
 head = newNode;  
 }  
 slow.prev = newNode;  
 }  
  
 void deleteFirstNode() {  
 if (head == null) {  
 System.*out*.println("List is already empty.");  
 return;  
 }  
 head = head.next;  
 if (head != null) {  
 head.prev = null;  
 } else {  
 tail = null;  
 }  
 }  
  
 void deleteMiddleNode() {  
 if (head == null || head.next == null) {  
 System.*out*.println("List is too small to delete the middle node.");  
 return;  
 }  
 Node14 slow = head, fast = head;  
 while (fast != null && fast.next != null) {  
 slow = slow.next;  
 fast = fast.next.next;  
 }  
 if (slow.prev != null) {  
 slow.prev.next = slow.next;  
 }  
 if (slow.next != null) {  
 slow.next.prev = slow.prev;  
 }  
 }  
  
 void deleteLastNode() {  
 if (tail == null) {  
 System.*out*.println("List is already empty.");  
 return;  
 }  
 tail = tail.prev;  
 if (tail != null) {  
 tail.next = null;  
 } else {  
 head = null;  
 }  
 }  
  
 boolean search(int key) {  
 Node14 temp = head;  
 while (temp != null) {  
 if (temp.data == key) {  
 return true;  
 }  
 temp = temp.next;  
 }  
 return false;  
 }  
  
 void display() {  
 Node14 temp = head;  
 while (temp != null) {  
 System.*out*.print("<--> " +temp.data );  
 temp = temp.next;  
 }  
 // System.out.println("NULL");  
 }  
}  
  
public class DoublyLinkedListMain14 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 DoublyLinkedList14 list = new DoublyLinkedList14();  
  
 System.*out*.print("Input the number of nodes: ");  
 int n = scanner.nextInt();  
  
 for (int i = 1; i <= n; i++) {  
 System.*out*.print("Input data for node " + i + ": ");  
 int data = scanner.nextInt();  
 list.insertAtEnd(data);  
 }  
  
 System.*out*.println("\nData entered in the list:");  
 list.display();  
  
// System.out.println("\nInserting at the beginning...");  
// list.insertAtBeginning(100);  
// list.display();  
//  
// System.out.println("\nInserting at the middle...");  
// list.insertAtMiddle(200);  
// list.display();  
//  
// System.out.println("\nDeleting the first node...");  
// list.deleteFirstNode();  
// list.display();  
//  
// System.out.println("\nDeleting the middle node...");  
// list.deleteMiddleNode();  
// list.display();  
//  
// System.out.println("\nDeleting the last node...");  
// list.deleteLastNode();  
// list.display();  
//  
// System.out.print("\nEnter an element to search: ");  
// int searchKey = scanner.nextInt();  
// if (list.search(searchKey)) {  
// System.out.println("Element found in the list.");  
// } else {  
// System.out.println("Element not found in the list.");  
// }  
 }  
}

**OUTPUT ->**

**Input the number of nodes: 5**

**Input data for node 1: 100**

**Input data for node 2: 200**

**Input data for node 3: 300**

**Input data for node 4: 400**

**Input data for node 5: 500**

**Data entered in the list:**

**<--> 100<--> 200<--> 300<--> 400<--> 500**

**13. Write a program in java to insert a new node at the beginning in a**

**doubly linked list.**

import java.util.Scanner;  
  
class Node14 {  
 int data;  
 Node14 next, prev;  
  
 Node14(int data) {  
 this.data = data;  
 this.next = null;  
 this.prev = null;  
 }  
}  
  
class DoublyLinkedList14 {  
 Node14 head, tail;  
  
 void insertAtBeginning(int data) {  
 Node14 newNode = new Node14(data);  
 if (head == null) {  
 head = tail = newNode;  
 } else {  
 newNode.next = head;  
 head.prev = newNode;  
 head = newNode;  
 }  
 }  
  
 void insertAtEnd(int data) {  
 Node14 newNode = new Node14(data);  
 if (head == null) {  
 head = tail = newNode;  
 return;  
 }  
 tail.next = newNode;  
 newNode.prev = tail;  
 tail = newNode;  
 }  
  
 void insertAtMiddle(int data) {  
 if (head == null) {  
 head = tail = new Node14(data);  
 return;  
 }  
 Node14 slow = head, fast = head;  
 while (fast != null && fast.next != null) {  
 slow = slow.next;  
 fast = fast.next.next;  
 }  
 Node14 newNode = new Node14(data);  
 newNode.next = slow;  
 newNode.prev = slow.prev;  
 if (slow.prev != null) {  
 slow.prev.next = newNode;  
 } else {  
 head = newNode;  
 }  
 slow.prev = newNode;  
 }  
  
 void deleteFirstNode() {  
 if (head == null) {  
 System.*out*.println("List is already empty.");  
 return;  
 }  
 head = head.next;  
 if (head != null) {  
 head.prev = null;  
 } else {  
 tail = null;  
 }  
 }  
  
 void deleteMiddleNode() {  
 if (head == null || head.next == null) {  
 System.*out*.println("List is too small to delete the middle node.");  
 return;  
 }  
 Node14 slow = head, fast = head;  
 while (fast != null && fast.next != null) {  
 slow = slow.next;  
 fast = fast.next.next;  
 }  
 if (slow.prev != null) {  
 slow.prev.next = slow.next;  
 }  
 if (slow.next != null) {  
 slow.next.prev = slow.prev;  
 }  
 }  
  
 void deleteLastNode() {  
 if (tail == null) {  
 System.*out*.println("List is already empty.");  
 return;  
 }  
 tail = tail.prev;  
 if (tail != null) {  
 tail.next = null;  
 } else {  
 head = null;  
 }  
 }  
  
 boolean search(int key) {  
 Node14 temp = head;  
 while (temp != null) {  
 if (temp.data == key) {  
 return true;  
 }  
 temp = temp.next;  
 }  
 return false;  
 }  
  
 void display() {  
 Node14 temp = head;  
 while (temp != null) {  
 System.*out*.print("<--> " +temp.data );  
 temp = temp.next;  
 }  
 // System.out.println("NULL");  
 }  
}  
  
public class DoublyLinkedListMain14 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 DoublyLinkedList14 list = new DoublyLinkedList14();  
  
 System.*out*.print("Input the number of nodes: ");  
 int n = scanner.nextInt();  
  
 for (int i = 1; i <= n; i++) {  
 System.*out*.print("Input data for node " + i + ": ");  
 int data = scanner.nextInt();  
 list.insertAtEnd(data);  
 }  
  
// System.out.println("\nData entered in the list:");  
// list.display();  
  
 System.*out*.println("\nInserting at the beginning...");  
 list.insertAtBeginning(100);  
 list.display();  
//  
// System.out.println("\nInserting at the middle...");  
// list.insertAtMiddle(200);  
// list.display();  
//  
// System.out.println("\nDeleting the first node...");  
// list.deleteFirstNode();  
// list.display();  
//  
// System.out.println("\nDeleting the middle node...");  
// list.deleteMiddleNode();  
// list.display();  
//  
// System.out.println("\nDeleting the last node...");  
// list.deleteLastNode();  
// list.display();  
//  
// System.out.print("\nEnter an element to search: ");  
// int searchKey = scanner.nextInt();  
// if (list.search(searchKey)) {  
// System.out.println("Element found in the list.");  
// } else {  
// System.out.println("Element not found in the list.");  
// }  
 }  
}

**OUTPUT ->**

**Input the number of nodes: 4**

**Input data for node 1: 200**

**Input data for node 2: 300**

**Input data for node 3: 400**

**Input data for node 4: 500**

**Inserting at the beginning...**

**<--> 100<--> 200<--> 300<--> 400<--> 500**

**15. Write a program in java to delete a node from the last of a doubly**

**linked list.**

import java.util.Scanner;  
  
class Node14 {  
 int data;  
 Node14 next, prev;  
  
 Node14(int data) {  
 this.data = data;  
 this.next = null;  
 this.prev = null;  
 }  
}  
  
class DoublyLinkedList14 {  
 Node14 head, tail;  
  
 void insertAtBeginning(int data) {  
 Node14 newNode = new Node14(data);  
 if (head == null) {  
 head = tail = newNode;  
 } else {  
 newNode.next = head;  
 head.prev = newNode;  
 head = newNode;  
 }  
 }  
  
 void insertAtEnd(int data) {  
 Node14 newNode = new Node14(data);  
 if (head == null) {  
 head = tail = newNode;  
 return;  
 }  
 tail.next = newNode;  
 newNode.prev = tail;  
 tail = newNode;  
 }  
  
 void insertAtMiddle(int data) {  
 if (head == null) {  
 head = tail = new Node14(data);  
 return;  
 }  
 Node14 slow = head, fast = head;  
 while (fast != null && fast.next != null) {  
 slow = slow.next;  
 fast = fast.next.next;  
 }  
 Node14 newNode = new Node14(data);  
 newNode.next = slow;  
 newNode.prev = slow.prev;  
 if (slow.prev != null) {  
 slow.prev.next = newNode;  
 } else {  
 head = newNode;  
 }  
 slow.prev = newNode;  
 }  
  
 void deleteFirstNode() {  
 if (head == null) {  
 System.*out*.println("List is already empty.");  
 return;  
 }  
 head = head.next;  
 if (head != null) {  
 head.prev = null;  
 } else {  
 tail = null;  
 }  
 }  
  
 void deleteMiddleNode() {  
 if (head == null || head.next == null) {  
 System.*out*.println("List is too small to delete the middle node.");  
 return;  
 }  
 Node14 slow = head, fast = head;  
 while (fast != null && fast.next != null) {  
 slow = slow.next;  
 fast = fast.next.next;  
 }  
 if (slow.prev != null) {  
 slow.prev.next = slow.next;  
 }  
 if (slow.next != null) {  
 slow.next.prev = slow.prev;  
 }  
 }  
  
 void deleteLastNode() {  
 if (tail == null) {  
 System.*out*.println("List is already empty.");  
 return;  
 }  
 tail = tail.prev;  
 if (tail != null) {  
 tail.next = null;  
 } else {  
 head = null;  
 }  
 }  
  
 boolean search(int key) {  
 Node14 temp = head;  
 while (temp != null) {  
 if (temp.data == key) {  
 return true;  
 }  
 temp = temp.next;  
 }  
 return false;  
 }  
  
 void display() {  
 Node14 temp = head;  
 while (temp != null) {  
 System.*out*.print("<--> " +temp.data );  
 temp = temp.next;  
 }  
 // System.out.println("NULL");  
 }  
}  
  
public class DoublyLinkedListMain14 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 DoublyLinkedList14 list = new DoublyLinkedList14();  
  
 System.*out*.print("Input the number of nodes: ");  
 int n = scanner.nextInt();  
  
 for (int i = 1; i <= n; i++) {  
 System.*out*.print("Input data for node " + i + ": ");  
 int data = scanner.nextInt();  
 list.insertAtEnd(data);  
 }  
  
// System.out.println("\nData entered in the list:");  
// list.display();  
  
// System.out.println("\nInserting at the beginning...");  
// list.insertAtBeginning(100);  
// list.display();  
//  
// System.out.println("\nInserting at the middle...");  
// list.insertAtMiddle(200);  
// list.display();  
//  
// System.out.println("\nDeleting the first node...");  
// list.deleteFirstNode();  
// list.display();  
//  
// System.out.println("\nDeleting the middle node...");  
// list.deleteMiddleNode();  
// list.display();  
//  
 System.*out*.println("\nDeleting the last node...");  
 list.deleteLastNode();  
 list.display();  
//  
// System.out.print("\nEnter an element to search: ");  
// int searchKey = scanner.nextInt();  
// if (list.search(searchKey)) {  
// System.out.println("Element found in the list.");  
// } else {  
// System.out.println("Element not found in the list.");  
// }  
 }  
}

**OUTPUT ->**

**Input the number of nodes: 4**

**Input data for node 1: 10**

**Input data for node 2: 20**

**Input data for node 3: 30**

**Input data for node 4: 40**

**Deleting the last node...**

**<--> 10<--> 20<--> 30**

**16. Write a program in java to create and display a circular linked**

**list.**

**Test Data :**

**Input the number of nodes : 3**

**Input data for node 1 : 2**

**Input data for node 2 : 5**

**Input data for node 3 : 8**

**Expected Output :**

**Data entered in the list are :**

**Data 1 = 2**

**Data 2 = 5**

**Data 3 = 8**

import java.util.Scanner;  
  
class Node15 {  
 int data;  
 Node15 next;  
  
 Node15(int data) {  
 this.data = data;  
 this.next = null;  
 }  
}  
  
class CircularLinkedList {  
 Node15 last;  
  
 void create(int n, Scanner scanner) {  
 for (int i = 1; i <= n; i++) {  
 System.*out*.print("Input data for node " + i + " : ");  
 int data = scanner.nextInt();  
 insert(data);  
 }  
 }  
  
 void insert(int data) {  
 Node15 newNode = new Node15(data);  
 if (last == null) {  
 last = newNode;  
 last.next = last;  
 } else {  
 newNode.next = last.next;  
 last.next = newNode;  
 last = newNode;  
 }  
 }  
  
 void display() {  
 if (last == null) {  
 return;  
 }  
 Node15 temp = last.next;  
 int count = 1;  
 do {  
 System.*out*.println("Data " + count + " = " + temp.data);  
 temp = temp.next;  
 count++;  
 } while (temp != last.next);  
 }  
}  
  
public class Main15 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 CircularLinkedList list = new CircularLinkedList();  
 System.*out*.print("Input the number of nodes : ");  
 int n = scanner.nextInt();  
 list.create(n, scanner);  
 System.*out*.println("Data entered in the list are :");  
 list.display();  
 scanner.close();  
 }  
}

**OUTPUT ->**

**Input the number of nodes : 4**

**Input data for node 1 : 10**

**Input data for node 2 : 20**

**Input data for node 3 : 30**

**Input data for node 4 : 40**

**Data entered in the list are :**

**Data 1 = 10**

**Data 2 = 20**

**Data 3 = 30**

**Data 4 = 40**

**17. Write a java programming to sort a given linked list.**

**Test Data and Expected Output : 5**

**15**

**33**

**49**

**6**

**65**

**Input number of elements in the linked list? Input the elements in the**

**linked list:**

**Sorted order is:**

**6 15 33 49 65**

import java.util.\*;  
  
class Node21 {  
 int data;  
 Node21 next;  
 Node21(int data) {  
 this.data = data;  
 this.next = null;  
 }  
}  
  
class linkedlist21 {  
 Node21 head;  
  
 void insert(int data) {  
 Node21 newNode = new Node21(data);  
 if (head == null) {  
 head = newNode;  
 return;  
 }  
 Node21 temp = head;  
 while (temp.next != null) {  
 temp = temp.next;  
 }  
 temp.next = newNode;  
 }  
  
 void sort() {  
 List<Integer> list = new ArrayList<>();  
 Node21 temp = head;  
 while (temp != null) {  
 list.add(temp.data);  
 temp = temp.next;  
 }  
 Collections.*sort*(list);  
 temp = head;  
 for (int num : list) {  
 temp.data = num;  
 temp = temp.next;  
 }  
 }  
  
 void display() {  
 Node21 temp = head;  
 while (temp != null) {  
 System.*out*.print(temp.data + " ");  
 temp = temp.next;  
 }  
 System.*out*.println();  
 }  
}  
  
public class SortLinkedList {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 linkedlist21 list = new linkedlist21();  
 int n = sc.nextInt();  
 for (int i = 0; i < n; i++) {  
 list.insert(sc.nextInt());  
 }  
 list.sort();  
 list.display();  
 }  
}

**OUTPUT ->**

**C:\Users\yadav\.jdks\openjdk-21.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.3.5\lib\idea\_rt.jar=53618:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.3.5\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath "C:\Users\yadav\OneDrive\Desktop\PRACTISE OF JAVA QUESTION\out\production\PRACTISE OF JAVA QUESTION" SortLinkedList**

**5**

**15**

**25**

**63**

**95**

**45**

**15 25 45 63 95**

**Q18. Write a java program to implement a stack using a singly linked**

**list.**

**DSA Assignment 1 (Stack, Queue, Linked List)**

**15**

**Expected Output:**

**Push data 1**

**Push data 2**

**Push data 3**

**Push data 4**

**Pop data: 4**

**Pop data: 3**

**Pop data: 2**

**Pop data: 1**

**Check a stack is empty or not?**

**Stack is empty!**

class StackNode {  
 int data;  
 StackNode next;  
 StackNode(int data) {  
 this.data = data;  
 this.next = null;  
 }  
}  
  
class Stack {  
 StackNode top;  
  
 void push(int data) {  
 StackNode newNode = new StackNode(data);  
 newNode.next = top;  
 top = newNode;  
 System.*out*.println("Push data " + data);  
 }  
  
 void pop() {  
 if (top == null) {  
 System.*out*.println("Stack is empty!");  
 return;  
 }  
 System.*out*.println("Pop data: " + top.data);  
 top = top.next;  
 }  
  
 boolean isEmpty() {  
 return top == null;  
 }  
  
 void insertAtEnd(int data) {  
 if (top == null) {  
 push(data);  
 return;  
 }  
 StackNode temp = top;  
 while (temp.next != null) {  
 temp = temp.next;  
 }  
 temp.next = new StackNode(data);  
 }  
  
 void display() {  
 StackNode temp = top;  
 while (temp != null) {  
 System.*out*.print(temp.data + " ");  
 temp = temp.next;  
 }  
 System.*out*.println();  
 }  
}  
  
public class StackLinkedList1 {  
 public static void main(String[] args) {  
 Stack stack = new Stack();  
 stack.push(1);  
 stack.push(2);  
 stack.push(3);  
 stack.push(4);  
 stack.pop();  
 stack.pop();  
 stack.pop();  
 stack.pop();  
 System.*out*.println("Check a stack is empty or not?");  
 if (stack.isEmpty()) {  
 System.*out*.println("Stack is empty!");  
 }  
 stack.insertAtEnd(22);  
 stack.display();  
 }  
}

**OUTPUT ->**

**Push data 1**

**Push data 2**

**Push data 3**

**Push data 4**

**Pop data: 4**

**Pop data: 3**

**Pop data: 2**

**Pop data: 1**

**Check a stack is empty or not?**

**Stack is empty!**

**Push data 22**

**22**