*HARSH KASHYAP  
CSE 4*

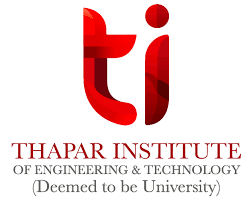
*101917088*

[*hkashyap\_be19@thapar.edu*](mailto:hkashyap_be19@thapar.edu)

A Practical activity Report submitted

for Data Structures (UCS301)

**DATA STRUCTURES**

****

Computer Science and Engineering

Patiala Campus

**2020**

Submitted to

Maninder Kaur

**Assignment 3**

**Question 1 using Circular**

**Develop a menu-driven program for the following operations of on a Circular as well as a Doubly Linked List.**

**(a) Insertion anywhere in the linked list (As a first node, as a last node, and after/before a specific node).**

**(b) Deletion of a specific node, say 'Delete Node 60'. That means the node to be deleted may appear as a head node, last node or a node in between.**

**(c) Search for a node.**

**SOLUTION CODE**

**/\*Develop a menu driven program for the following operations of on a Circular as well**

**as a Doubly Linked List.**

**(a) Insertion anywhere in the linked list (As a first node, as a last node, and**

**after/before a specific node).**

**(b) Deletion of a specific node, say 'Delete Node 60'. That mean the node to be**

**deleted may appear as a head node, last node or a node in between.**

**(c) Search for a node.**

**\*/**

**import java.util.\*;**

**import java.lang.\*;**

**class CircularLinkedList**

**{**

**Node head;**

**static class Node**

**{**

**Node next;**

**int data;**

**Node(int d)**

**{**

**data =d;**

**next =null;**

**}**

**}**

**static CircularLinkedList insertAtBeginning(CircularLinkedList l, int d)**

**{**

**Node cons=l.head;**

**Node fresh= new Node(d);**

**if (l.head==null)**

**{**

**l.head=fresh;**

**fresh.next=l.head;**

**}**

**else**

**{**

**Node curr =l.head;**

**do**

**{**

**curr=curr.next;**

**}while(curr.next!=cons);**

**curr.next=fresh;**

**fresh.next=l.head;**

**l.head=fresh;**

**}**

**return l;**

**}**

**public static CircularLinkedList insertAtEnd(CircularLinkedList list, int data)**

**{**

**Node new\_Node=new Node(data);**

**new\_Node.next=null;**

**if (list.head==null)**

**{**

**list.head = new\_Node;**

**new\_Node.next=list.head;**

**}**

**else**

**{**

**Node constant = list.head;**

**Node last = list.head;**

**do**

**{**

**last = last.next;**

**}while(last.next!= constant);**

**last.next = new\_Node;**

**new\_Node.next=list.head;**

**}**

**return list;**

**}**

**static CircularLinkedList insertAfter(CircularLinkedList l, int d, int v)**

**{**

**boolean f=false;**

**Node fresh= new Node(d);**

**Node curr=l.head;**

**if (l.head==null)**

**{**

**System.out.println("linkedList is null, so can't insert");**

**return null;**

**}**

**else**

**{**

**do**

**{**

**if (curr.data == v && curr.next!=l.head)**

**{**

**f=true;**

**fresh.next=curr.next;**

**curr.next=fresh;**

**break;**

**}**

**else if (curr.data == v && curr.next==l.head)**

**{**

**f=true;**

**fresh.next=l.head;**

**curr.next=fresh;**

**break;**

**}**

**curr=curr.next;**

**}while(curr.next!=l.head);**

**}**

**if(!f)**

**System.out.println("Didn't insert , "+v+" not found.");**

**return l;**

**}**

**//Delete element from key**

**public static CircularLinkedList deleteAtKey(CircularLinkedList list, int key)**

**{**

**Node current = list.head;**

**Node constant = list.head;**

**Node prev = null;**

**if (current.data==key)**

**{**

**list.head = current.next;**

**System.out.println(key +" found and deleted ,was present at the very beginning ");**

**do**

**{**

**current = current.next;**

**}while (current.next!=constant);**

**current.next=list.head;**

**return list;**

**}**

**do**

**{**

**if (current.data==key)**

**{**

**prev.next=current.next;**

**System.out.println(key +" found and deleted. ");**

**return list;**

**}**

**prev=current;**

**current=current.next;**

**}while(current!=constant);**

**System.out.println(key +" not found. ");**

**return list;**

**}**

**static void print(CircularLinkedList l)**

**{**

**Node cons=l.head;**

**Node curr=l.head;**

**if (curr==null)**

**{**

**System.out.print("No element found. ");**

**return;**

**}**

**System.out.print("-> ");**

**do**

**{**

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

**curr=curr.next;**

**}while(curr!=cons);**

**}**

**public static void search(CircularLinkedList list, int val)**

**{**

**Node current = list.head;**

**Node constant = list.head;**

**do**

**{**

**if (current.data == val)**

**{**

**System.out.println(val +" is present.");**

**return;**

**}**

**current = current.next;**

**}while (current!=constant);**

**System.out.println(val+" not found in list. ");**

**}**

**// child class which can use basic operation of parent class**

**//TO USE THIS PLEASE CALL THE CLASS CircularLinkedList TOO**

**static Scanner scr= new Scanner(System.in);**

**static int menu()**

**{**

**System.out.println( );**

**System.out.println( "--------——CIRCULAR LINKED LIST----------");**

**System.out.println( "\t——MENU——- \n1.Insertion at the beginning.\n2.Insertion at the end.\n3.Insert after ");**

**System.out.println( "4.Deletion of a specific node");**

**System.out.println( "5.Search for a node.\n6.Display all node values.\n7.Exit \nEnter your choice ");**

**int ch = scr.nextInt();**

**return ch;**

**}**

**//main function**

**public static void main(String[] args)**

**{**

**CircularLinkedList list = new CircularLinkedList();**

**int val;**

**int ch=0;**

**do {**

**ch = menu();**

**System.out.println();**

**switch (ch)**

**{**

**case 1:**

**System.out.print("Enter value to be inserted at the beginning : ");**

**val =scr.nextInt();**

**insertAtBeginning(list, val);**

**break;**

**case 2:**

**System.out.print("Enter value to be inserted at the end : ");**

**val =scr.nextInt();**

**insertAtEnd(list, val);**

**break;**

**case 3:**

**System.out.print("Enter value to be inserted : ");**

**val =scr.nextInt();**

**System.out.print(val+" to be inserted after which node : ");**

**int no =scr.nextInt();**

**insertAfter(list, val, no);**

**break;**

**case 4:**

**System.out.print("Enter value to be deleted : ");**

**val =scr.nextInt();**

**deleteAtKey(list,val);**

**break;**

**case 5:**

**System.out.print("Enter value to be searched : ");**

**val =scr.nextInt();**

**search(list, val);**

**break;**

**case 6:**

**print(list);**

**break;**

**case 7:**

**System.out.println("We are done ");**

**break;**

**default:**

**System.out.println("Not an option \n");**

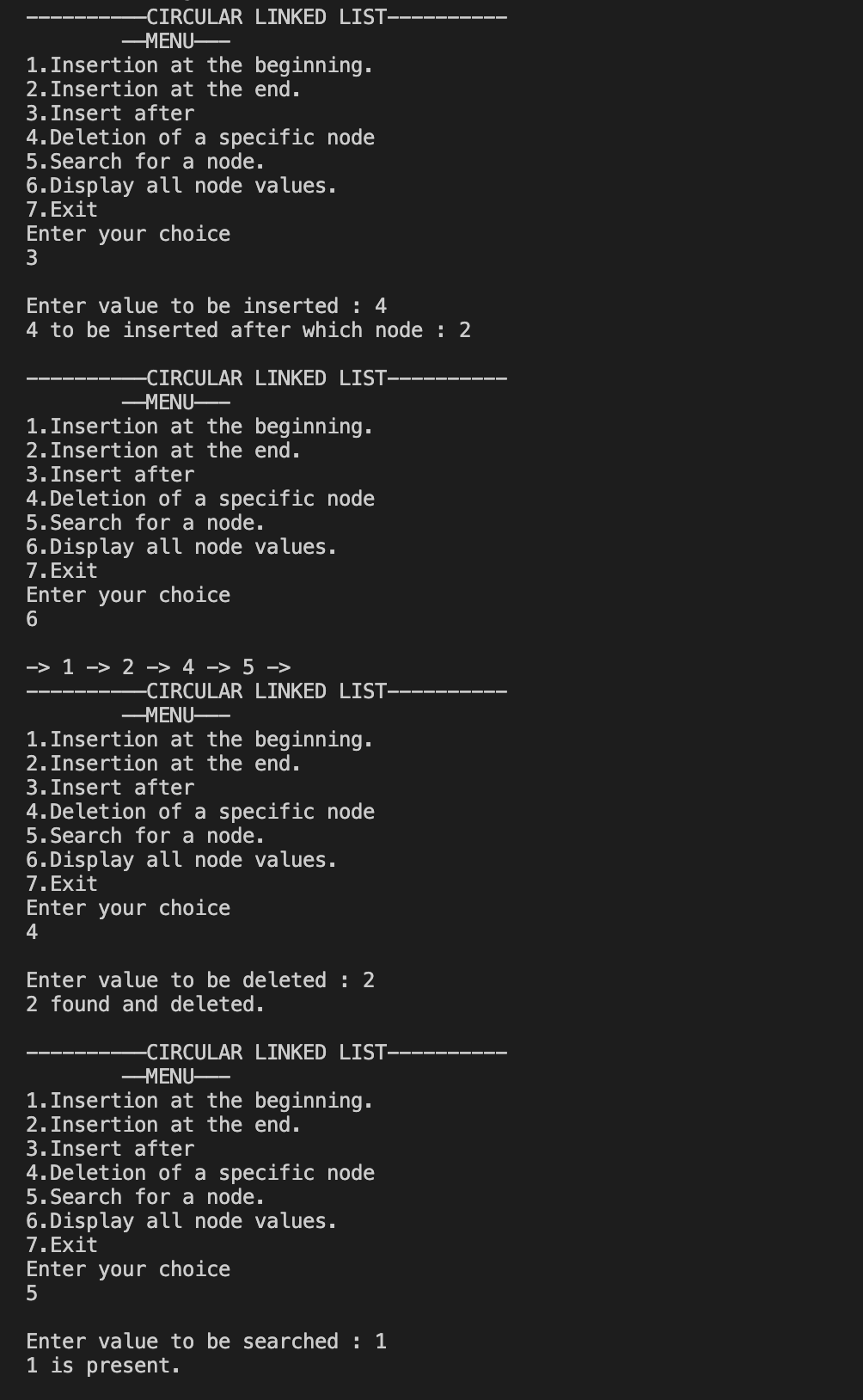
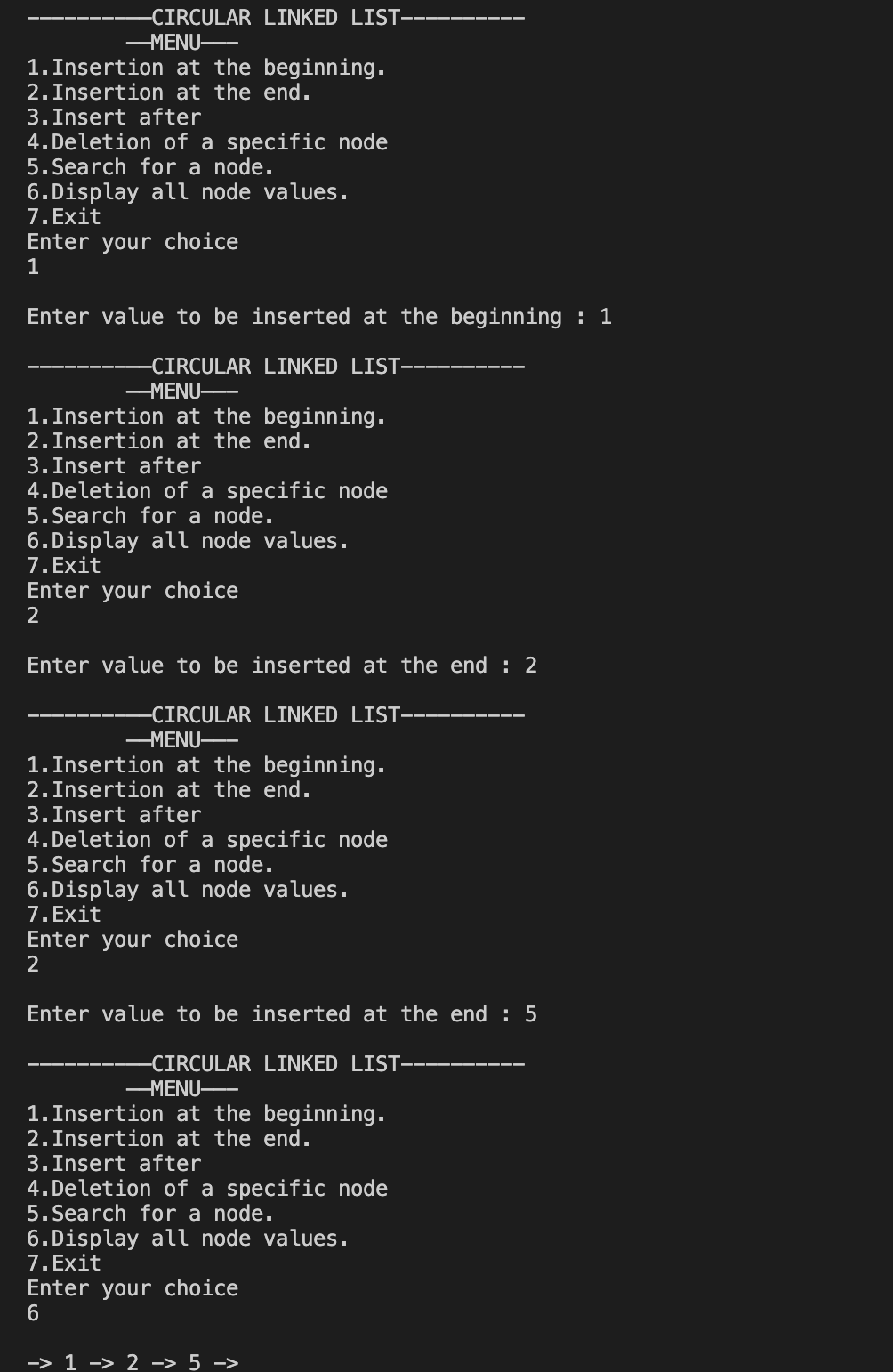
**}**

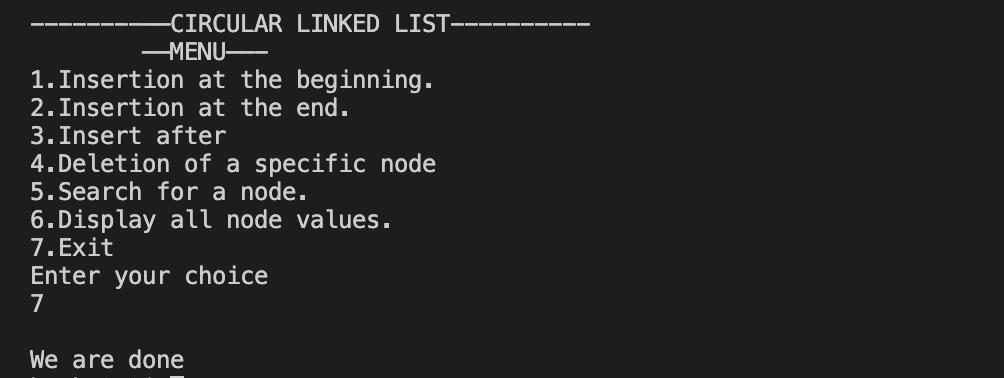
**} while (ch != 7);**

**}**

**}**

**OUTPUT**





**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 1 using Doubly Linked List**

**Develop a menu-driven program for the following operations of on a Circular as well as a Doubly Linked List.**

**(a) Insertion anywhere in the linked list (As a first node, as a last node, and after/before a specific node).**

**(b) Deletion of a specific node, say 'Delete Node 60'. That means the node to be deleted may appear as a head node, last node or a node in between.**

**(c) Search for a node.**

**SOLUTION CODE**

**import java.util.\*;**

**class DoubleLinkedList**

**{**

**Node head;**

**static class Node**

**{**

**int data;**

**Node prev;**

**Node next;**

**Node(int d)**

**{**

**data=d;**

**prev=null;**

**next=null;**

**}**

**}**

**static DoubleLinkedList insertAtBeginning(DoubleLinkedList dll, int d)**

**{**

**Node fresh= new Node(d);**

**fresh.prev=null;**

**fresh.next=null;**

**if (dll.head==null)**

**{**

**dll.head=fresh;**

**return dll;**

**}**

**else**

**{**

**fresh.next=dll.head;**

**dll.head.prev=fresh;**

**dll.head=fresh;**

**}**

**return dll;**

**}**

**static DoubleLinkedList insertAtEnd(DoubleLinkedList dll, int d)**

**{**

**Node fresh= new Node(d);**

**fresh.prev=null;**

**fresh.next=null;**

**if (dll.head==null)**

**{**

**dll.head=fresh;**

**}**

**else**

**{**

**Node last =dll.head;**

**while(last.next!=null)**

**{**

**last=last.next;**

**}**

**last.next=fresh;**

**fresh.prev=last;**

**}**

**return dll;**

**}**

**//printing**

**static void print(DoubleLinkedList dll)**

**{**

**System.out.println("Printing Doubly Linked List : ");**

**Node last = dll.head;**

**while(last!=null)**

**{**

**if(last.next!=null)**

**{**

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

**}**

**else**

**{**

**System.out.print(last.data+" ");**

**}**

**last=last.next;**

**}**

**}**

**static DoubleLinkedList insertAfter(DoubleLinkedList dll, int d, int s)**

**{**

**Node fresh= new Node(d);**

**if(dll.head==null)**

**{**

**System.out.println("Cannot insert in an empty linked list");**

**return dll;**

**}**

**Node last=dll.head;**

**if(dll.head.data==s)**

**{**

**fresh.next=dll.head.next;**

**dll.head.next=fresh;**

**fresh.prev=dll.head;**

**return dll;**

**}**

**while(last!=null)**

**{**

**if (last.data==s)**

**{**

**fresh.next=last.next;**

**last.next=fresh;**

**fresh.prev=last;**

**return dll;**

**}**

**last=last.next;**

**}**

**System.out.println("Number not found, didn't insert");**

**return dll;**

**}**

**static DoubleLinkedList insertBefore(DoubleLinkedList dll, int d, int s)**

**{**

**Node fresh= new Node(d);**

**if(dll.head==null)**

**{**

**System.out.println("Cannot insert in an empty linked list");**

**return dll;**

**}**

**Node last=dll.head;**

**if(dll.head.data==s)**

**{**

**insertAtBeginning(dll, d);**

**}**

**while(last!=null)**

**{**

**if (last.data==s)**

**{**

**fresh.next=last;**

**fresh.prev=last.prev;**

**(last.prev).next=fresh;**

**return dll;**

**}**

**last=last.next;**

**}**

**System.out.println("Number not found, didn't insert");**

**return dll;**

**}**

**static DoubleLinkedList deleteAtKey(DoubleLinkedList dll, int d)**

**{**

**if(dll.head==null)**

**{**

**System.out.println("Cannot delete in an empty linked list");**

**return dll;**

**}**

**Node last=dll.head;**

**Node previous=dll.head;**

**if(dll.head.data==d)**

**{**

**dll.head=dll.head.next;**

**dll.head.prev=null;**

**return dll;**

**}**

**while(last!=null)**

**{**

**if (last.data==d && last.next!=null)**

**{**

**previous.next=last.next;**

**(last.next).prev=previous;**

**return dll;**

**}**

**if (last.data==d && last.next==null)**

**{**

**previous.next=last.next;**

**return dll;**

**}**

**previous=last;**

**last=last.next;**

**}**

**System.out.println("Number not found, didn't delete");**

**return dll;**

**}**

**static void search(DoubleLinkedList dll, int d)**

**{**

**if(dll.head==null)**

**{**

**System.out.println("Cannot search in an empty linked list");**

**return ;**

**}**

**Node last=dll.head;**

**int i=1;**

**while(last!=null)**

**{**

**if (last.data==d )**

**{**

**System.out.println(d+" found at index "+i);**

**return;**

**}**

**i++;**

**last=last.next;**

**}**

**System.out.println(d+" not found");**

**return ;**

**}**

**static Scanner scr= new Scanner(System.in);**

**static int menu()**

**{**

**System.out.println( );**

**System.out.println( "--------——DOUBLY LINKED LIST----------");**

**System.out.println( "\t——MENU——- \n1.Insertion at the beginning.\n2.Insertion at the end.\n3.Insert after ");**

**System.out.println( "4.Insert Before \n5.Deletion of a specific node");**

**System.out.println( "6.Search for a node.\n7.Display all node values.\n8.Exit \nEnter your choice ");**

**int ch = scr.nextInt();**

**return ch;**

**}**

**//main function**

**public static void main(String[] args)**

**{**

**DoubleLinkedList list = new DoubleLinkedList();**

**int val;**

**int ch=0;**

**do {**

**ch = menu();**

**System.out.println();**

**switch (ch)**

**{**

**case 1:**

**System.out.print("Enter value to be inserted at the beginning : ");**

**val =scr.nextInt();**

**insertAtBeginning(list, val);**

**break;**

**case 2:**

**System.out.print("Enter value to be inserted at the end : ");**

**val =scr.nextInt();**

**insertAtEnd(list, val);**

**break;**

**case 3:**

**System.out.print("Enter value to be inserted : ");**

**val =scr.nextInt();**

**System.out.print(val+" to be inserted after which node : ");**

**int no =scr.nextInt();**

**insertAfter(list, val, no);**

**break;**

**case 4:**

**System.out.print("Enter value to be inserted : ");**

**val =scr.nextInt();**

**System.out.print(val+" to be inserted before which node : ");**

**no =scr.nextInt();**

**insertBefore(list, val, no);**

**break;**

**case 5:**

**System.out.print("Enter value to be deleted : ");**

**val =scr.nextInt();**

**deleteAtKey(list,val);**

**break;**

**case 6:**

**System.out.print("Enter value to be searched : ");**

**val =scr.nextInt();**

**search(list, val);**

**break;**

**case 7:**

**print(list);**

**break;**

**case 8:**

**System.out.println("We are done ");**

**break;**

**default:**

**System.out.println("Not an option \n");**

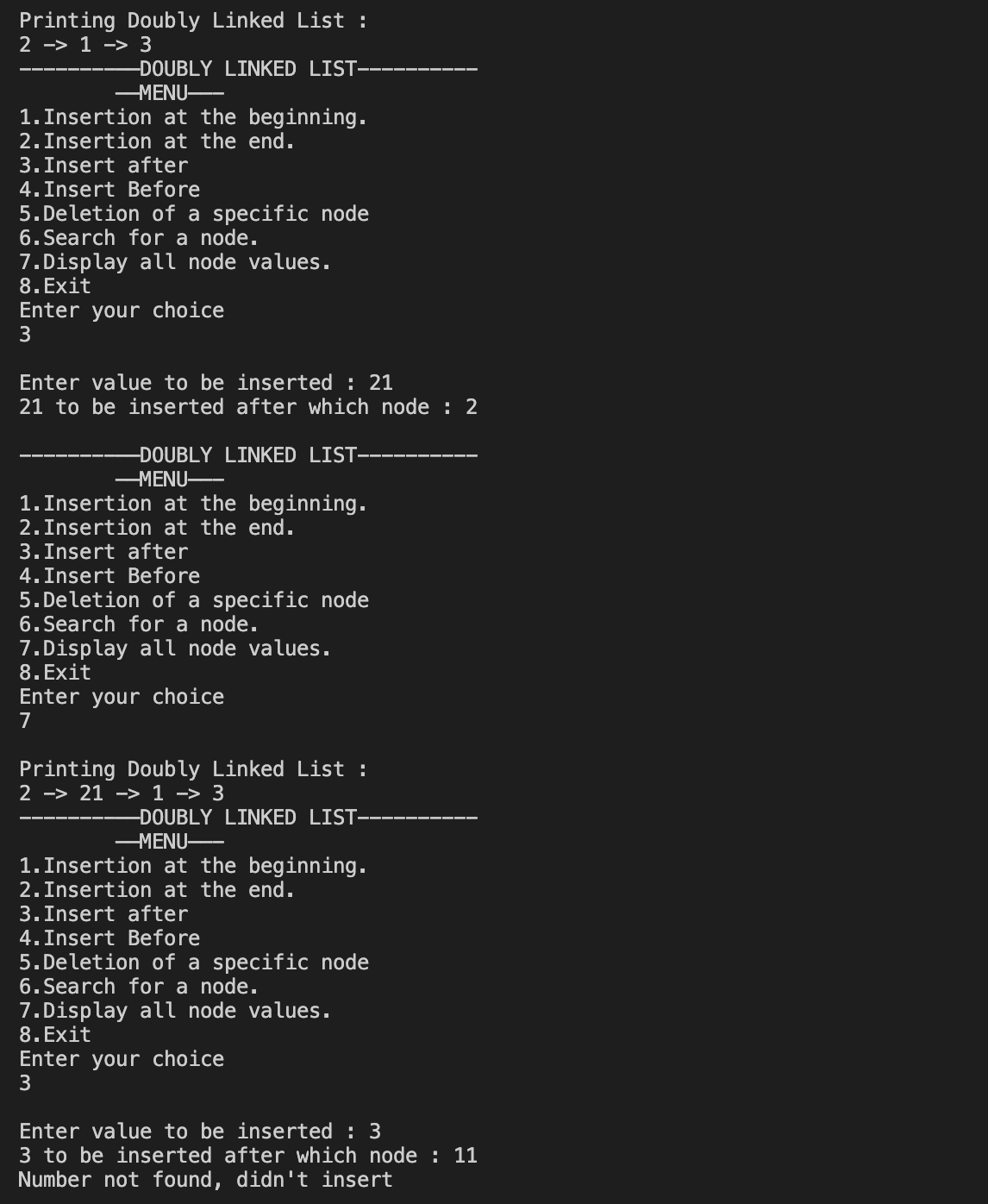
**}**

**} while (ch != 8);**

**}**

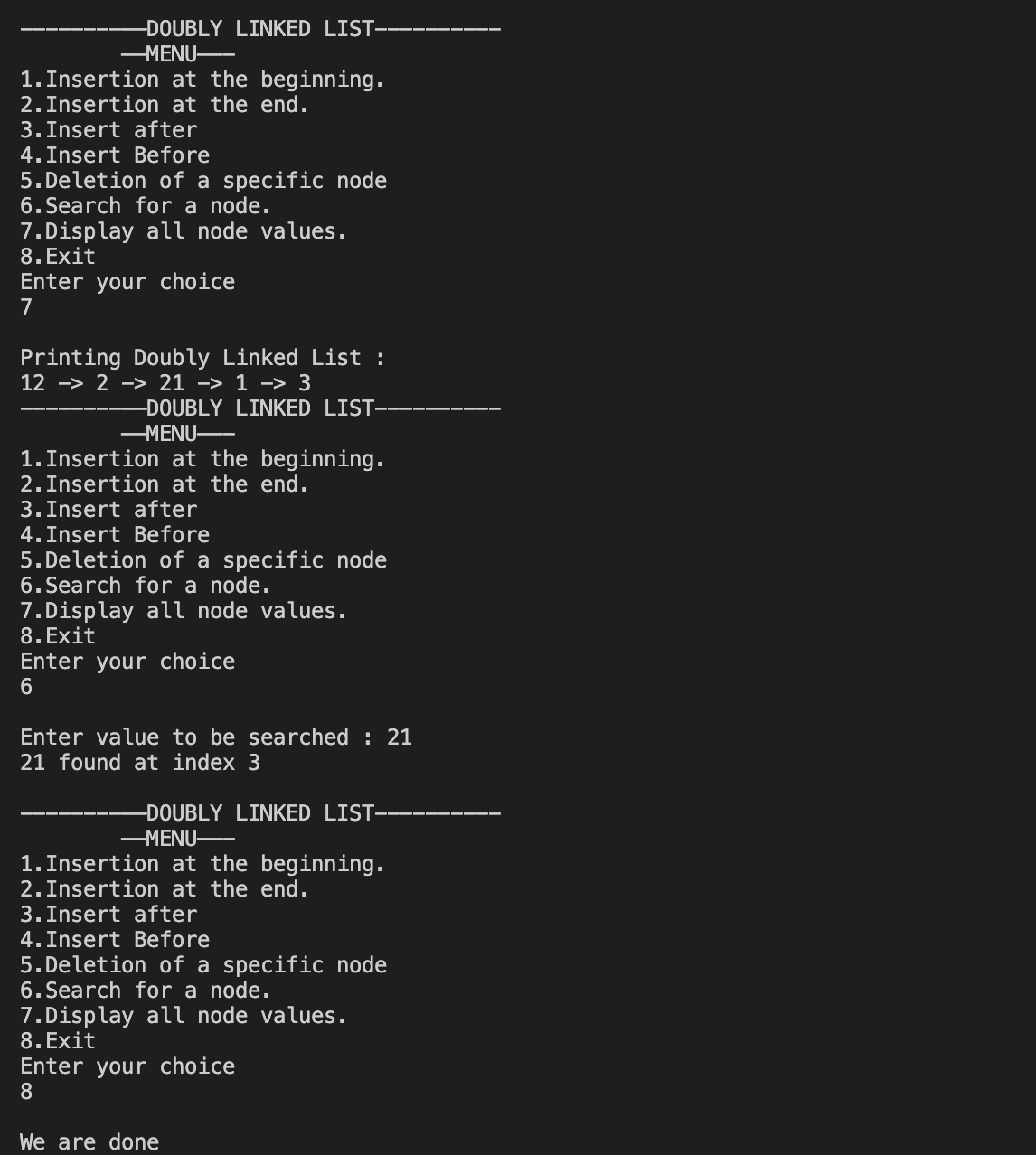
**}**

**OUTPUT**







****

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 2**

**Display all the node values in a circular linked list, repeating value of head node at the end too. For example, if elements present in the circular linked list starting from head are 20 → 100 → 40 → 80 → 60, then output should be displayed as 20 100 40 80 60 20.**

**SOLUTION CODE**

**/\*Display all the node values in a circular linked list, repeating value of head node at the**

**end too. For example, if elements present in the circular linked list starting from head**

**are 20 → 100 → 40 → 80 → 60, then output should be displayed as 20 100 40 80 60**

**20.**

**\*/**

**import java.util.\*;**

**class Node**

**{**

**int data;**

**Node next;**

**Node(int d)**

**{**

**data=d;**

**next=null;**

**}**

**}**

**class a3ques2**

**{**

**static Scanner scr= new Scanner(System.in);**

**static Node insert(int n)**

**{**

**Node last=new Node(-1);**

**for (int i=0;i<n;i++)**

**{**

**System.out.print("Enter value: ");**

**int data=scr.nextInt();**

**Node tmp=new Node(data);**

**if(last.data==-1)**

**{**

**last=tmp;**

**tmp.next=tmp;**

**}**

**else**

**{**

**tmp.next=last.next;**

**last.next=tmp;**

**last=tmp;**

**}**

**}**

**return last;**

**}**

**//main function**

**public static void main(String[] args)**

**{**

**System.out.println("\nEnter number of elements: ");**

**int n=scr.nextInt();**

**Node last =insert(n);**

**Node cl=last.next;**

**do**

**{**

**System.out.print(cl.data+" ");**

**cl=cl.next;**

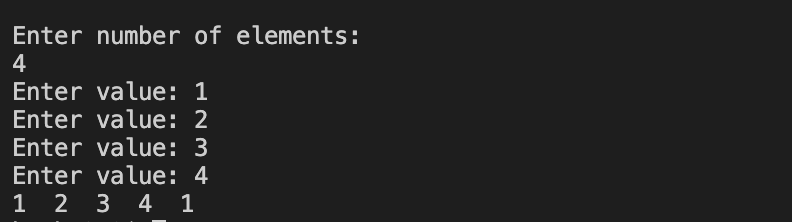
**}while(cl!=last.next);**

**System.out.println(cl.data+" ");**

**}**

**}**

**OUTPUT**

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 3**

**Write a program to find the size of**

**a) Doubly Linked List.**

**b) Circular Linked List.**

**SOLUTION CODE for a) Doubly Linked List**

**// WAP to find size of Double Linked List**

**import java.util.\*;**

**class DLinkedList**

**{**

**Node head;**

**static class Node**

**{**

**int data;**

**Node prev;**

**Node next;**

**Node(int d)**

**{**

**data=d;**

**prev=null;**

**next=null;**

**}**

**}**

**static Scanner scr= new Scanner(System.in);**

**public static void insert(DLinkedList dll)**

**{**

**while(true)**

**{**

**System.out.print("Enter element or 0 to stop : ");**

**int data=scr.nextInt();**

**if(data==0)**

**{**

**System.out.println("Stopping the insertion process");**

**break;**

**}**

**insertAtEnd(dll,data);**

**}**

**}**

**static DLinkedList insertAtEnd(DLinkedList dll, int d)**

**{**

**Node fresh= new Node(d);**

**fresh.prev=null;**

**fresh.next=null;**

**if (dll.head==null)**

**{**

**dll.head=fresh;**

**}**

**else**

**{**

**Node last =dll.head;**

**while(last.next!=null)**

**{**

**last=last.next;**

**}**

**last.next=fresh;**

**fresh.prev=last;**

**}**

**return dll;**

**}**

**//main function**

**public static void main(String[] args)**

**{**

**DLinkedList dll= new DLinkedList();**

**System.out.println("Taking input... ");**

**insert(dll);**

**int k=1;**

**Node curr=dll.head;**

**while(curr!=null)**

**{**

**k++;**

**curr=curr.next;**

**}**

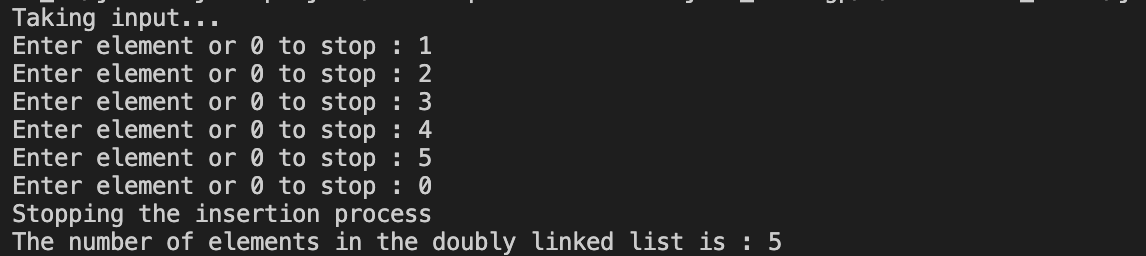
**k--;**

**System.out.println("The number of elements in the doubly linked list is : "+k);**

**}**

**}**

***OUTPUT***

******

**SOLUTION CODE for b) Circular Linked List**

**// WAP to find size of Circular Linked List**

**import java.util.\*;**

**// child class which can use basic operation of parent class**

**//TO USE THIS PLEASE CALL THE CLASS CircularLinkedList TOO**

**import java.util.\*;**

**class Node**

**{**

**int data;**

**Node next;**

**Node(int d)**

**{**

**data=d;**

**next=null;**

**}**

**}**

**class a3ques3b**

**{**

**static Scanner scr= new Scanner(System.in);**

**static Node insert()**

**{**

**Node last=new Node(-1);**

**while(true)**

**{**

**System.out.print("Enter element or 0 to stop : ");**

**int data=scr.nextInt();**

**Node tmp=new Node(data);**

**if(data==0)**

**{**

**System.out.println("Stopping the insertion process");**

**break;**

**}**

**if(last.data==-1)**

**{**

**last=tmp;**

**tmp.next=tmp;**

**}**

**else**

**{**

**tmp.next=last.next;**

**last.next=tmp;**

**last=tmp;**

**}**

**}**

**return last;**

**}**

**//main function**

**public static void main(String[] args)**

**{**

**Node last = insert();**

**int k=1;**

**Node curr=last.next;**

**do**

**{**

**k++;**

**curr=curr.next;**

**}while(curr!=last.next);**

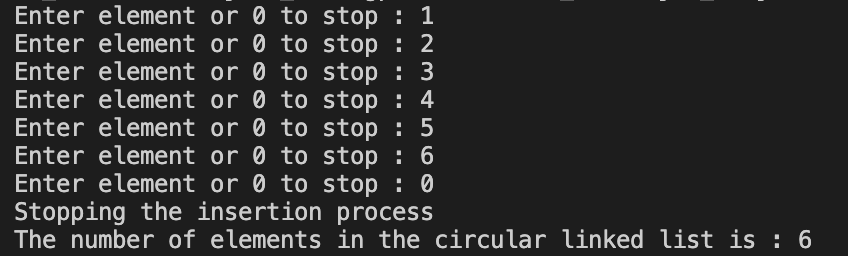
**k--;**

**System.out.println("The number of elements in the circular linked list is : "+k);**

**}**

**}**

***OUTPUT***

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 4**

**Check if a doubly-linked list of characters is palindrome or not**

**SOLUTION CODE**

**// WAP to check if a double Linked List is plaindrome**

**// WAP to find size of Double Linked List**

**import java.util.\*;**

**class DLList**

**{**

**Node head;**

**static class Node**

**{**

**int data;**

**Node prev;**

**Node next;**

**Node(int d)**

**{**

**data=d;**

**prev=null;**

**next=null;**

**}**

**}**

**static Scanner scr= new Scanner(System.in);**

**public static void insert(DLList dll)**

**{**

**while(true)**

**{**

**System.out.print("Enter element or 0 to stop : ");**

**char data=scr.next().charAt(0);**

**if(data=='0')**

**{**

**System.out.println("Stopping the insertion process");**

**break;**

**}**

**insertAtEnd(dll,data);**

**}**

**}**

**static DLList insertAtEnd(DLList dll, int d)**

**{**

**Node fresh= new Node(d);**

**fresh.prev=null;**

**fresh.next=null;**

**if (dll.head==null)**

**{**

**dll.head=fresh;**

**}**

**else**

**{**

**Node last =dll.head;**

**while(last.next!=null)**

**{**

**last=last.next;**

**}**

**last.next=fresh;**

**fresh.prev=last;**

**}**

**return dll;**

**}**

**static boolean isPalindrome(DLList dll)**

**{**

**Node left=dll.head;**

**Node right=dll.head;**

**while(right.next!=null)**

**right=right.next;**

**while(left!=right)**

**{**

**if(left.data!=right.data)**

**return false;**

**left=left.next;**

**right=right.prev;**

**}**

**return true;**

**}**

**//main function**

**public static void main(String[] args)**

**{**

**DLList dll= new DLList();**

**System.out.println("Taking input... ");**

**insert(dll);**

**boolean ans= isPalindrome(dll);**

**if(ans)**

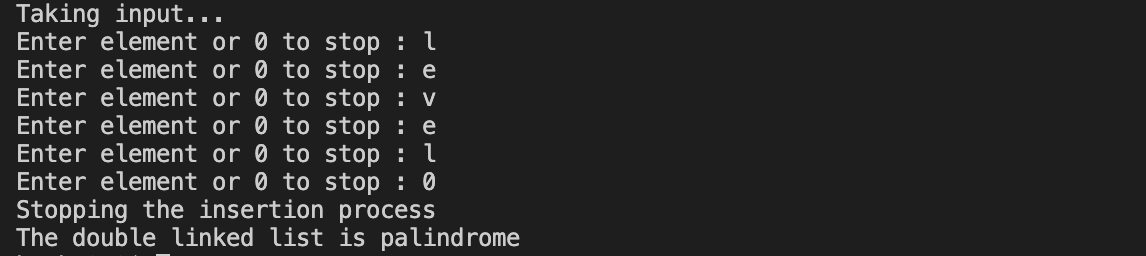
**System.out.println("The double linked list is palindrome");**

**else**

**System.out.println("The double linked list is not palindrome");**

**}**

**}**

***OUTPUT***

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 5**

**Write a program to check if a linked list is Circular Linked List or not.**

**SOLUTION CODE**

**//WAP to Check if a linked list is Circular Linked List**

**import java.util.\*;**

**class Node**

**{**

**Node next;**

**int data;**

**Node(int d)**

**{**

**data=d;**

**next=null;**

**}**

**static Scanner scr= new Scanner(System.in);**

**public static void main(String[] args)**

**{**

**System.out.println("Enter 0 for termination");**

**System.out.print("Enter element : ");**

**int value=scr.nextInt();**

**if (value==0)**

**{**

**System.out.println("This is a circular linked list");**

**return;**

**}**

**Node head=new Node(value);**

**Node last = new Node(value);**

**while(true)**

**{**

**System.out.print("Enter element : ");**

**value=scr.nextInt();**

**if (value==0)**

**{**

**System.out.println("Ending insertion process...");**

**break;**

**}**

**Node ins=new Node(value);**

**last.next=ins;**

**last=ins;**

**ins.next=null;**

**}**

**System.out.println("Enter 0 to make linked list circular, 1 to make it a single linked list.");**

**int ch=scr.nextInt();**

**if (ch==0)**

**last.next=head;**

**//checking**

**System.out.println("Checking if it is circular or not.... ");**

**if (last.next==null)**

**{**

**System.out.println("Not a Circular Linked List ");**

**}**

**else**

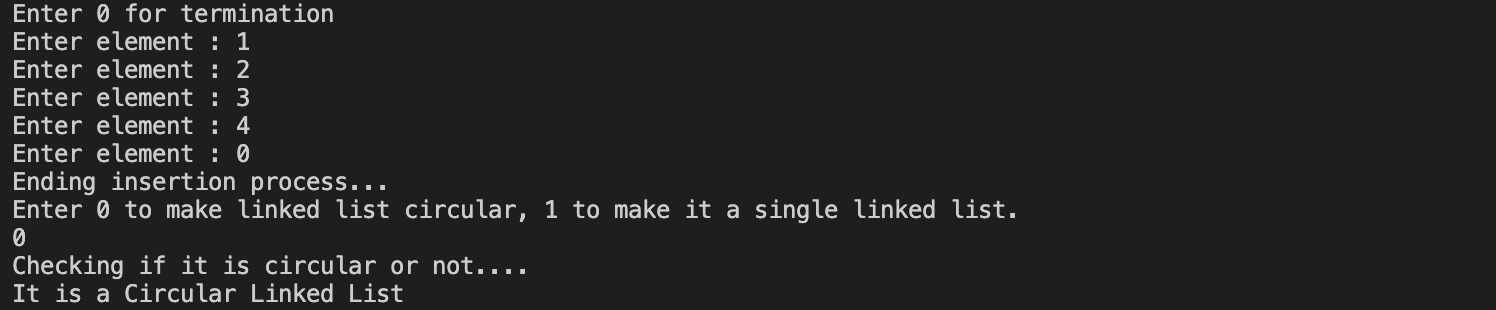
**{**

**System.out.println("It is a Circular Linked List ");**

**}**

**}**

**}**

***OUTPUT***

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**EXTRA**

**Question**

**WAP to Split a Circular Linked List into two halves.**

**SOLUTION CODE**

**// WAP to Split a Circular Linked List into two halves**

**import java.util.\*;**

**class Node**

**{**

**int data;**

**Node next;**

**Node(int d)**

**{**

**data=d;**

**next=null;**

**}**

**}**

**class splitcircular**

**{**

**static Scanner scr= new Scanner(System.in);**

**static Node insert()**

**{**

**Node last=new Node(-1);**

**while(true)**

**{**

**System.out.print("Enter element or 0 to stop : ");**

**int data=scr.nextInt();**

**Node tmp=new Node(data);**

**if(data==0)**

**{**

**System.out.println("Stopping the insertion process");**

**break;**

**}**

**if(last.data==-1)**

**{**

**last=tmp;**

**tmp.next=tmp;**

**}**

**else**

**{**

**tmp.next=last.next;**

**last.next=tmp;**

**last=tmp;**

**}**

**}**

**return last;**

**}**

**static void print(Node end)**

**{**

**Node copy=end.next;**

**do**

**{**

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

**copy=copy.next;**

**}while(copy!=end.next);**

**System.out.println(" -> ");**

**}**

**static Node insertAtEnd(int d,int k, Node last)**

**{**

**Node fresh=new Node(d);**

**if (k==1) //counter to checkmif it is a first element**

**{**

**last=fresh;**

**fresh.next=fresh;**

**}**

**else**

**{**

**fresh.next=last.next;**

**last.next=fresh;**

**last=fresh;**

**}**

**return last;**

**}**

**//main function**

**public static void main(String[] args)**

**{**

**Node last=insert();**

**System.out.println("\nThe circular linked list is :");**

**print(last);**

**Node slow =last.next;**

**Node fast =last.next;**

**Node divLast1 = new Node(0);**

**Node divLast2 = new Node(0);**

**int k=1;**

**while(fast.next!=last.next && fast.next.next!=last.next)**

**{**

**divLast1=insertAtEnd(slow.data,k++,divLast1);**

**slow=slow.next;**

**fast=fast.next.next;**

**}**

**divLast1= insertAtEnd(slow.data,k++,divLast1);**

**k=1;**

**slow=slow.next;**

**while(slow!=last.next)**

**{**

**divLast2=insertAtEnd(slow.data,k++,divLast2);**

**slow=slow.next;**

**}**

**System.out.println("\nThe splitted linked list is : \n1." );**

**print(divLast1);**

**System.out.println("\n2." );**

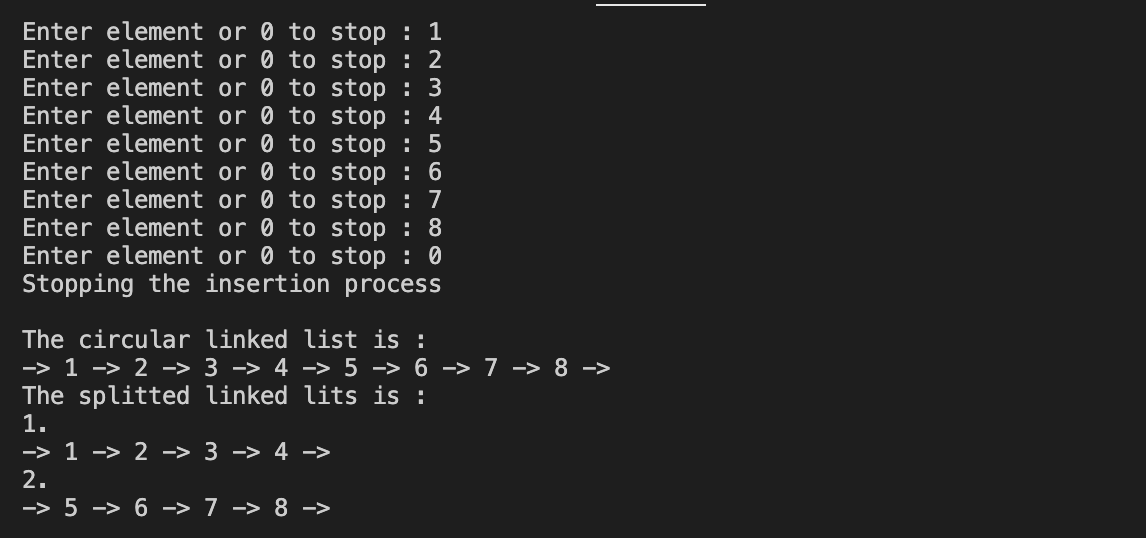
**print(divLast2);**

**System.out.println("\n" );**

**}**

**}**

***OUTPUT***

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question**

**WAP to remove all even parity nodes from a Doubly and Circular Singly Linked List**

**SOLUTION CODE**

**//Remove all even parity nodes from a Doubly and Circular Singly Linked List**

**import java.util.\*;**

**class Parity**

**{**

**static class Node**

**{**

**Node next;**

**int data;**

**Node(int d)**

**{**

**next=null;**

**data=d;**

**}**

**}**

**static int i;**

**static Node last;**

**static Scanner scr= new Scanner(System.in);**

**static Node insert(Node last,int d)**

**{**

**Node fresh= new Node(d);**

**fresh.next=last.next;**

**last.next=fresh;**

**last=fresh;**

**return last;**

**}**

**static void input()**

**{**

**i=1;**

**System.out.println("Keep on inserting the values, enter 0 to end");**

**while(true)**

**{**

**System.out.print("Value : ");**

**int d=scr.nextInt();**

**if(d==0)**

**{**

**System.out.println("Breaking the process.");**

**break;**

**}**

**if (i++==1)**

**{**

**last= new Node(d);**

**last.next=last;**

**}**

**else**

**last=insert(last, d);**

**}**

**}**

**static void print(Node last)**

**{**

**Node curr=last.next;**

**if (curr==null)**

**{**

**System.out.print("No element found. ");**

**return;**

**}**

**System.out.print("-> ");**

**do**

**{**

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

**curr=curr.next;**

**}while(curr!=last.next);**

**}**

**static boolean getParity(int n)**

**{**

**boolean parity = false;**

**while(n != 0)**

**{**

**parity = !parity;**

**n = n & (n-1);**

**}**

**return !parity;**

**}**

**public static void main(String[] args)**

**{**

**input();**

**print(last); //head**

**Node curr=last.next;**

**Node prev=last;**

**for(int j=1;j<i-1;j++)**

**{**

**if(getParity(curr.data))**

**{**

**prev.next=curr.next;**

**curr=curr.next;**

**}**

**else**

**{**

**prev=curr;**

**curr=curr.next;**

**}**

**}**

**System.out.println("\nLinkedList with even parity is : ");**

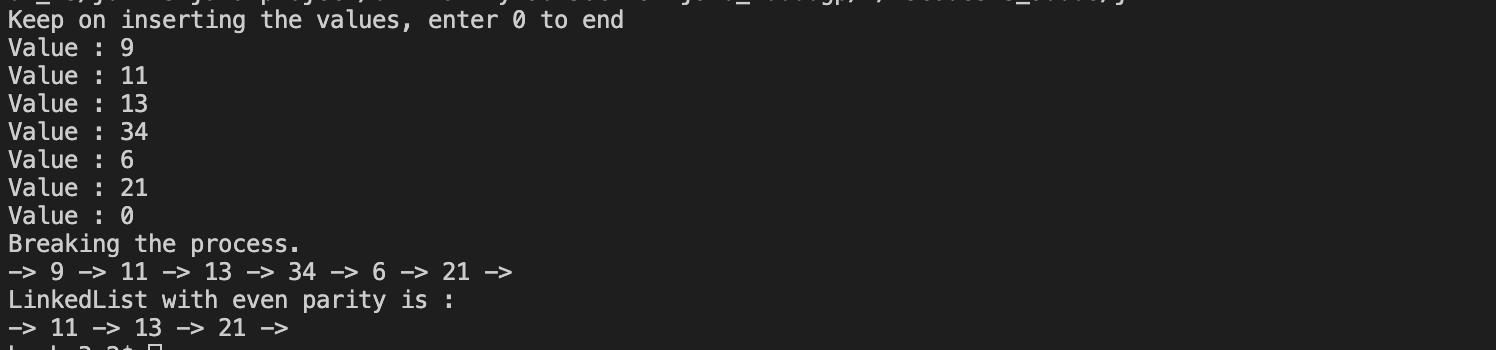
**print(last);**

**System.out.println();**

**}**

**}**

**OUTPUT**

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question**

**WAP to Construct a Doubly linked list from 2D Matrix**

**SOLUTION CODE**

**// Java program to construct a Doubly linked list from 2D Matrix**

**import java.util.\*;**

**class DD**

**{**

**// define dimension of matrix**

**static int dim;**

**static class Node**

**{**

**int data;**

**Node next;**

**Node prev;**

**Node up;**

**Node down;**

**};**

**static Scanner scr= new Scanner(System.in);**

**static int[][] matrixInput()**

**{**

**System.out.print("Enter number of rows and columns (NXN) : ");**

**int n=scr.nextInt();**

**int arr[][]=new int[n][n];**

**for (int i=0;i<n;i++)**

**{**

**for(int j=0;j<n;j++)**

**{**

**System.out.print("Enter ( "+i+","+j+" ) value : ");**

**arr[i][j]=scr.nextInt();**

**}**

**}**

**return arr;**

**}**

**// function to create a new node**

**static Node createNode(int data)**

**{**

**Node temp = new Node();**

**temp.data = data;**

**temp.next = null;**

**temp.prev = null;**

**temp.up = null;**

**temp.down = null;**

**return temp;**

**}**

**// function to construct the**

**// doubly linked list**

**static Node constructDoublyListUtil(int mtrx[][],int i, int j,Node curr)**

**{**

**if (i >= dim || j >= dim) {**

**return null;**

**}**

**Node temp = createNode(mtrx[i][j]);**

**temp.prev = curr;**

**temp.up = curr;**

**temp.next= constructDoublyListUtil(mtrx, i, j + 1, temp);**

**temp.down= constructDoublyListUtil(mtrx, i + 1, j, temp);**

**return temp;**

**}**

**// Function to construct the doubly linked list**

**static Node constructDoublyList(int mtrx[][])**

**{**

**return constructDoublyListUtil(mtrx, 0, 0, null);**

**}**

**// function for displaying**

**// doubly linked list data**

**static void display(Node head)**

**{**

**Node column;**

**Node row= head;**

**while (row != null)**

**{**

**column = row;**

**while (column!=null)**

**{**

**System.out.print(column.data+" ");**

**column = column.next;**

**}**

**System.out.print("\n");**

**row = row.down;**

**}**

**}**

**// driver code**

**public static void main(String args[])**

**{**

**// initialise matrix**

**int mtrx[][] = matrixInput();**

**dim = mtrx.length;**

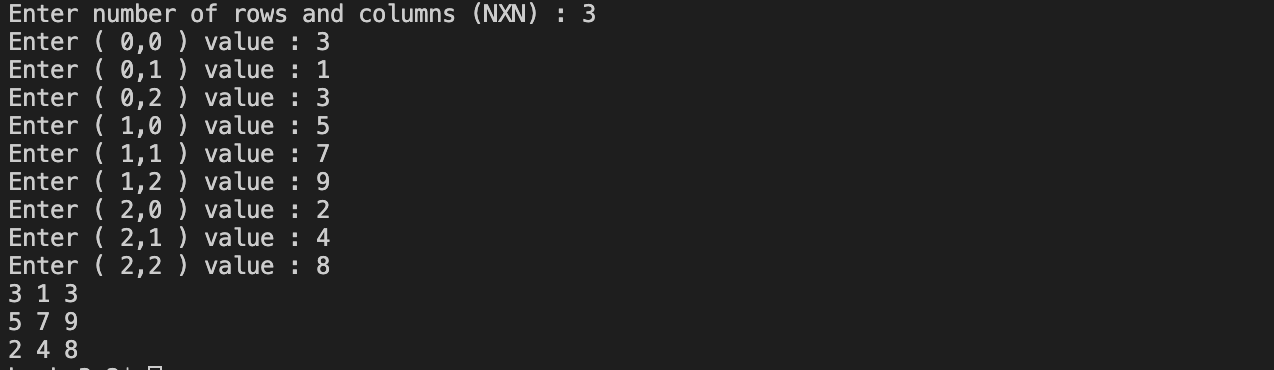
**Node list = constructDoublyList(mtrx);**

**display(list);**

**}**

**}**

***OUTPUT***

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**