LAB ASSIGNMENT NO – 3

Qn1 . Write a Java program to create a double linked list and Following Functionalities must by implement in the program.

Insert a New node 100 at the end of the list
Insert a new node 200 at the end of the list
Insert a new node 300 at the end of the list
Display the list
Insert a new node 400 after the last node
Display the list
Delete 400 from the list
Delete 400 from the list
Display the list
Delete 200 from the list
Display the current elements present in the list

```java
package com.doublylinkedlistdemo.main;

import com.doublylinkedlistdemo.ops.DoublyLinkedListDemo;

public class AppMain {
    Run | Debug
    public static void main(String[] args) {
        DoublyLinkedListDemo list = new DoublyLinkedListDemo();

        list.insertAtEnd(data:100);
        list.insertAtEnd(data:200);
        list.insertAtEnd(data:300);

        System.out.println("Current elements in the list:");
        list.displayList();

        list.insertAfterLastNode(data:400);
        System.out.println("Updated list after inserting 400:");
        list.displayList();

        list.deleteNode(data:400);
        list.deleteNode(data:400); // Attempt to delete 400 again
        System.out.println("Updated list after deleting 400:");
        list.displayList();

        list.deleteNode(data:200);
        System.out.println("Current elements present in the list:");
        list.displayList();
    }
}
```

```java
package com.doublylinkedlistdemo.ops;

public class DoublyLinkedListDemo {

    private Node head;
    private Node tail;

    private class Node {
        int data;
        Node prev;
        Node next;

        public Node(int data) {
            this.data = data;
            this.prev = null;
            this.next = null;
        }
    }

    public void insertAtEnd(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
            tail = newNode;
        } else {
            newNode.prev = tail;
            tail.next = newNode;
            tail = newNode;
        }
    }
```

```java
    public void insertAfterLastNode(int data) {
        if (tail == null) {
            System.out.println("List is empty. Cannot insert after the last node.");
            return;
        }
        Node newNode = new Node(data);
        newNode.prev = tail;
        tail.next = newNode;
        tail = newNode;
    }

    public void deleteNode(int data) {
        Node current = head;
        while (current != null) {
            if (current.data == data) {
                if (current == head) {
                    head = current.next;
                    if (head != null) {
                        head.prev = null;
                    } else {
                        tail = null; // List becomes empty
                    }
                } else {
                    current.prev.next = current.next;
                    if (current != tail) {
                        current.next.prev = current.prev;
                    } else {
                        tail = current.prev;
                    }
                }
                System.out.println("Deleted " + data + " from the list.");
                return;
            }
            current = current.next;
        }
        System.out.println("Node with data " + data + " not found in the list.");
    }
```

```java
    public void displayList() {
        Node current = head;
        while (current != null) {
            System.out.print(current.data + " ");
            current = current.next;
        }
        System.out.println();
    }
}
```

```
nisha@nisha-Cloud:/media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/DSA/day3$  /usr/bin/env /u
sr/lib/jvm/java-8-openjdk-amd64/jre/bin/java -cp /home/nisha/.config/Code/User/workspaceSt
orage/5f96979cdc32d6617db6296c98238c4b/redhat.java/jdt_ws/day3_694147cc/bin com.doublylink
edlistdemo.main.AppMain
Current elements in the list:
100 200 300
Updated list after inserting 400:
100 200 300 400
Deleted 400 from the list.
Node with data 400 not found in the list.
Updated list after deleting 400:
100 200 300
Deleted 200 from the list.
Current elements present in the list:
100 300
```

Q2 . Write a program to display numbers from 1 to n (in any order)
Using recursion [tail & head].

```java
package com.recursiondemo.head;

public class HeadRecursionDemo {
    Run | Debug
    public static void main(String[] args) {
        int n = 5;
        headRecursion(n);
    }

    static void headRecursion(int n) {
        if (n == 0) {
            return;
        }
        headRecursion(n - 1); // Recursive call before printing
        System.out.print(n + " ");
    }
}
```

```
/nisha/.config/Code/User/workspaceStorage/19c4a3ec
/bin com.recursiondemo.head.HeadRecursionDemo
1 2 3 4 5 nisha@nisha-Cloud:/media/sf_Vertual_Box_
```

```java
package com.recursiondemo.tail;

public class TailRecursionDemo {
    Run | Debug
    public static void main(String[] args) {
        int n = 5;
        tailRecursion(n, current:1);
    }

    static void tailRecursion(int n, int current) {
        if (current > n) {
            return;
        }
        System.out.print(current + " "); // Print before the recursive call
        tailRecursion(n, current + 1);
    }
}
```

```
/lib/jvm/java-8-openjdk-amd64/jre/bin/j
ava -cp /home/nisha/.config/Code/User/w
orkspaceStorage/19c4a3ed9dd1b6ebec750a3
d6a29b3ca/redhat.java/jdt_ws/day32_bee7
b1e6/bin com.recursiondemo.tail.TailRec
ursionDemo
1 2 3 4 5 nisha@nisha-Cloud:/media/sf_V
```

Q3 . Write a java program to find Sum of First 10 even Numbers using any recursion Technique

```java
package com.recursiondemo.main;

public class EvenSumHeadRecursive {
    Run | Debug
    public static void main(String[] args) {
        int n = 10;
        int sum = calculateSumOfFirstNEvens(n);
        System.out.println("The sum of the first " + n + " even numbers is: " + sum);
    }

    static int calculateSumOfFirstNEvens(int n) {
        if (n == 1) {
            return 2; // Base case: The first even number is 2.
        }

        int currentEven = 2 * n;
        int previousSum = calculateSumOfFirstNEvens(n - 1);

        return currentEven + previousSum;
    }
}
```

```
nisha@nisha-Cloud:/media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/DSA/day3 3$  /usr/bin/env /usr/lib/jvm/java-8-
openjdk-amd64/jre/bin/java -cp /home/nisha/.config/Code/User/workspaceStorage/f292cb6959db40e773a820e71b1a2a6e
/redhat.java/jdt_ws/day3\ 3_1e0e88df/bin com.recursiondemo.main.EvenSumHeadRecursive
The sum of the first 10 even numbers is: 110
```

Q4 . Write a java program to find factorial of a number using any recursion Technique

```java
package com.recursiondemo.tail;

public class TailRecursionFactorialDemo {
    Run | Debug
    public static void main(String[] args) {
        int n = 5;
        int factorial = calculateFactorialTailRecursion(n, result:1);
        System.out.println("Factorial of " + n + " is: " + factorial);
    }

    static int calculateFactorialTailRecursion(int n, int result) {
        if (n == 0) {
            return result;
        }
        return calculateFactorialTailRecursion(n - 1, n * result);
    }
}
```

```
nisha@nisha-Cloud:/media/sf_Vertual_Box_Share/Nisha_Ubuntu/Cdac/DSA/day3 4$  cd /media/sf_Vertual_Bo
 ; /usr/bin/env /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java -cp /home/nisha/.config/Code/User/wor
fb03e91ed/redhat.java/jdt_ws/day3\ 4_1e0e88e0/bin com.recursiondemo.tail.TailRecursionFactorialDemo
Factorial of 5 is: 120
```