

**UNIVERSITY INSTITUTE OF ENGINEERING**

**Department of Computer Science & Engineering**

**Subject Name:** Competitive Coding

**Subject Code:** 20CSP-314

**Submitted to: Submitted by:**

Er. Mamta Punia Name: Sahil Kaundal

UID: 21BCS8197

Section: 616

Group: A

**INDEX**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Ex. No** | **List of Experiments** | **Conduct (MM: 12)** | **Viva**  **(MM: 10)** | **Record (MM: 8)** | **Total**  **(MM: 30)** | **Remarks/Signature** |
| 1 | To demonstrate the concept of Array. |  |  |  |  |  |
| 2 | To demonstrate the concept of Stack and Queue. |  |  |  |  |  |
| 3 | **To demonstrate the concept of Linked List.** |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**Experiment 3.1**

**Student Name:** Sahil Kaundal **UID:** 21BCS8197

**Branch:** BE CSE (Lateral Entry) **Section/Group:** 616/A

**Semester:** 5th **Date of Performance:** 04/09/2022

**Subject Name:** CC Lab **Subject Code:** 20CSP-314

1. **Aim/Overview of the practical:**

Given a reference to the head of a doubly-linked list and an integer, data, create a new DoublyLinkedListNode object having data value data and insert it at the proper location to maintain the sort.

1. **Apparatus / Simulator Used:**

* Windows 7 or above
* Google Chrome

**3. Code:**

import java.io.\*;

import java.math.\*;

import java.security.\*;

import java.text.\*;

import java.util.\*;

import java.util.concurrent.\*;

import java.util.regex.\*;

public class Solution {

    static class DoublyLinkedListNode {

        public int data;

        public DoublyLinkedListNode next;

        public DoublyLinkedListNode prev;

        public DoublyLinkedListNode(int nodeData) {

            this.data = nodeData;

            this.next = null;

            this.prev = null;

        }

    }

    static class DoublyLinkedList {

        public DoublyLinkedListNode head;

        public DoublyLinkedListNode tail;

        public DoublyLinkedList() {

            this.head = null;

            this.tail = null;

        }

        public void insertNode(int nodeData) {

            DoublyLinkedListNode node = new DoublyLinkedListNode(nodeData);

            if (this.head == null) {

                this.head = node;

            } else {

                this.tail.next = node;

                node.prev = this.tail;

            }

            this.tail = node;

        }

    }

    public static void printDoublyLinkedList(DoublyLinkedListNode node, String sep, BufferedWriter bufferedWriter) throws IOException {

        while (node != null) {

            bufferedWriter.write(String.valueOf(node.data));

            node = node.next;

            if (node != null) {

                bufferedWriter.write(sep);

            }

        }

    }

 static DoublyLinkedListNode sortedInsert(DoublyLinkedListNode head, int data) {

        DoublyLinkedListNode node = new DoublyLinkedListNode(data);

        if (head == null) {

            return node;

        } else if (data <= head.data) {

            node.next = head;

            head.prev = node;

            return node;

        } else {

            DoublyLinkedListNode ptr = sortedInsert(head.next, data);

            head.next = ptr;

            ptr.prev = head;

            return head;

        }

    }

    private static final Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) throws IOException {

        BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

        int t = scanner.nextInt();

        scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

        for (int tItr = 0; tItr < t; tItr++) {

            DoublyLinkedList llist = new DoublyLinkedList();

            int llistCount = scanner.nextInt();

            scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

            for (int i = 0; i < llistCount; i++) {

                int llistItem = scanner.nextInt();

                scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

                llist.insertNode(llistItem);

            }

            int data = scanner.nextInt();

            scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

            DoublyLinkedListNode llist1 = sortedInsert(llist.head, data);

            printDoublyLinkedList(llist1, " ", bufferedWriter);

            bufferedWriter.newLine();

        }

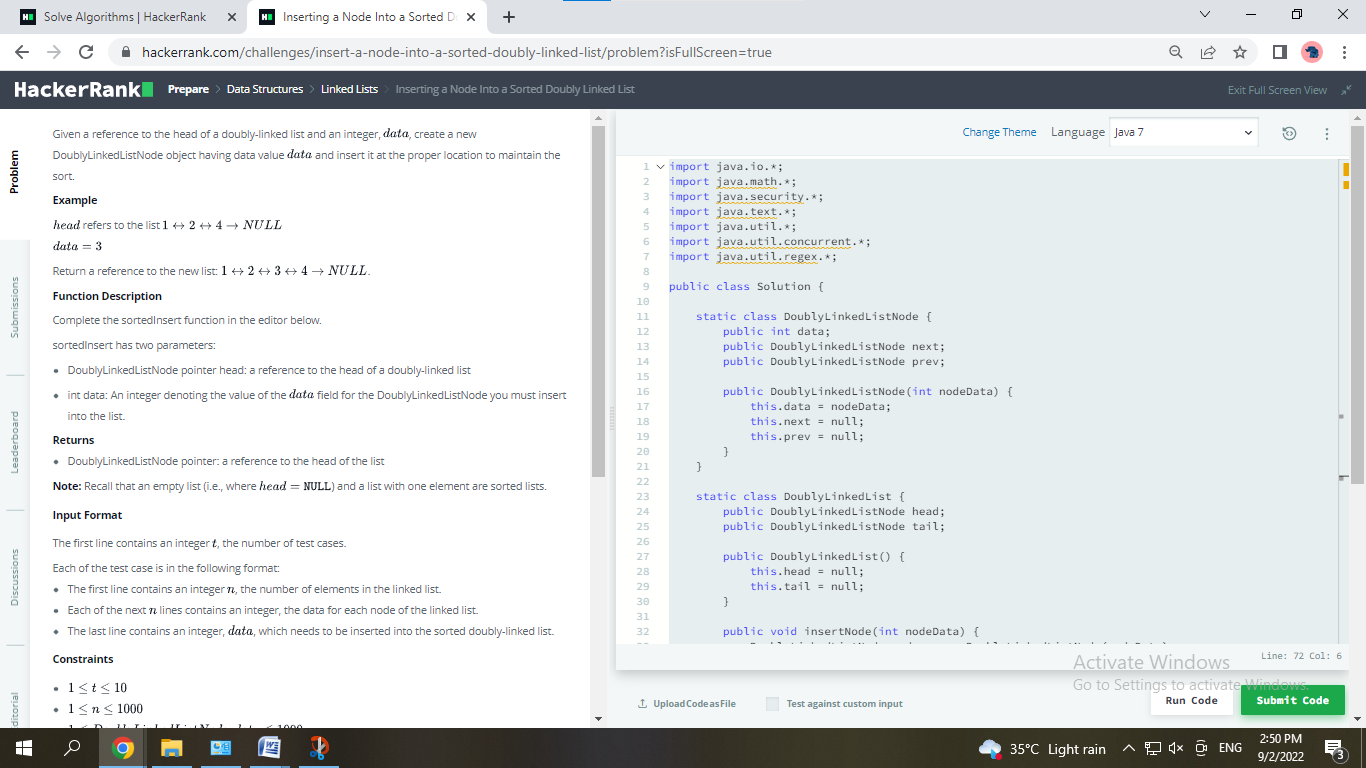
        bufferedWriter.close();

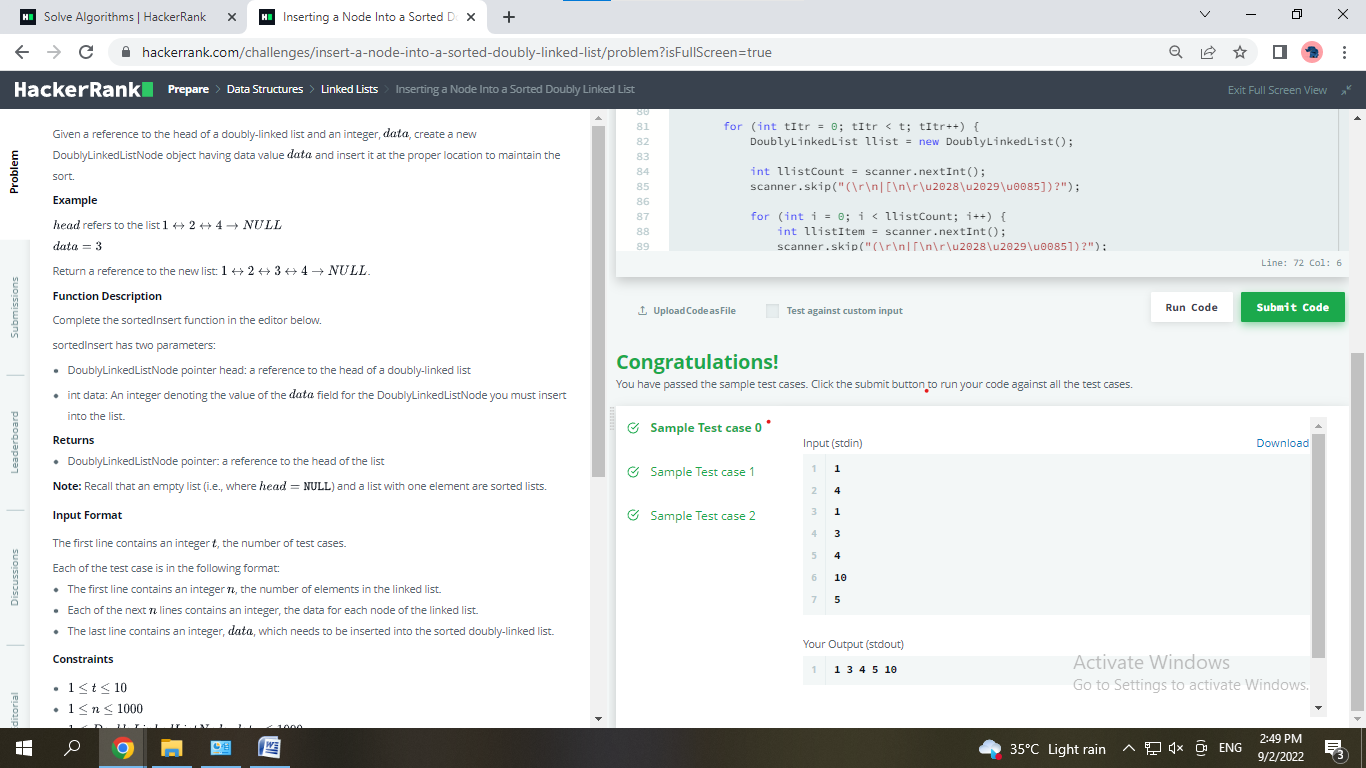
        scanner.close();

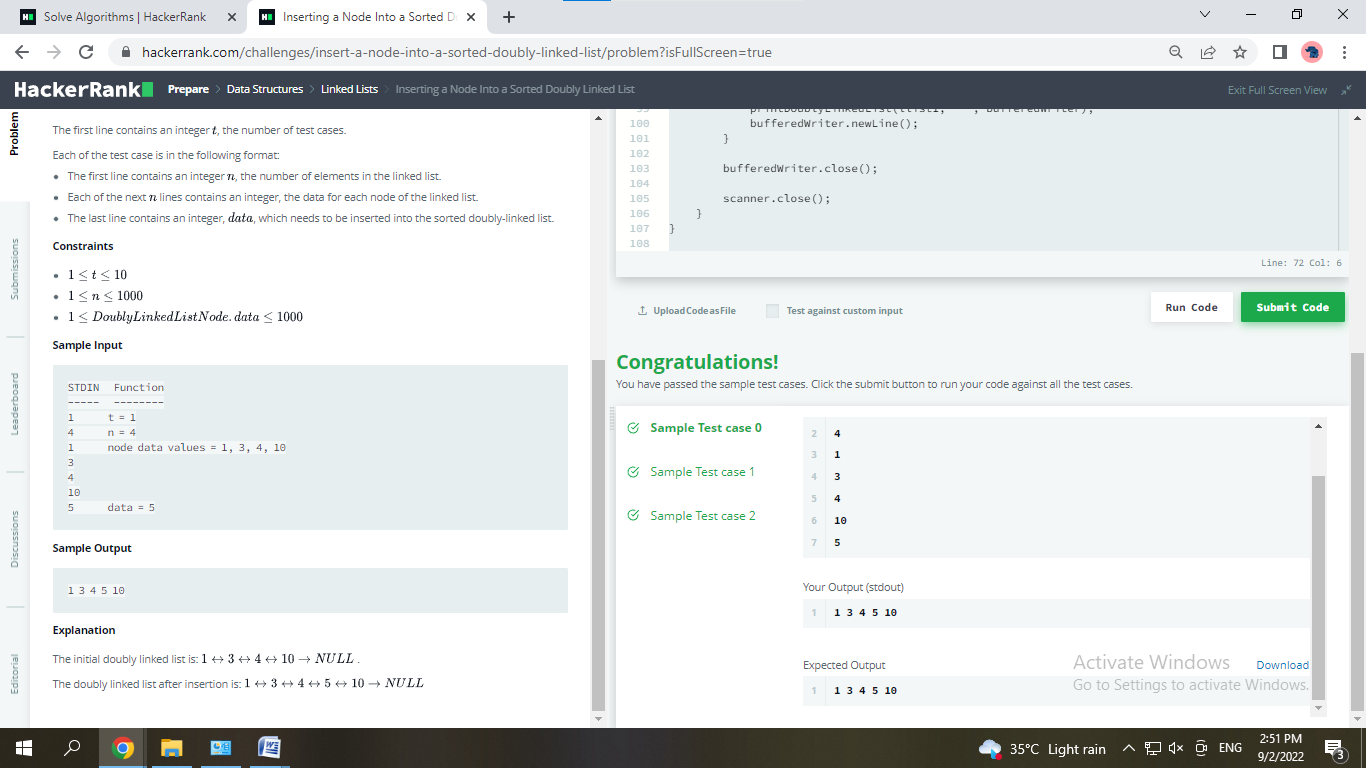
    }

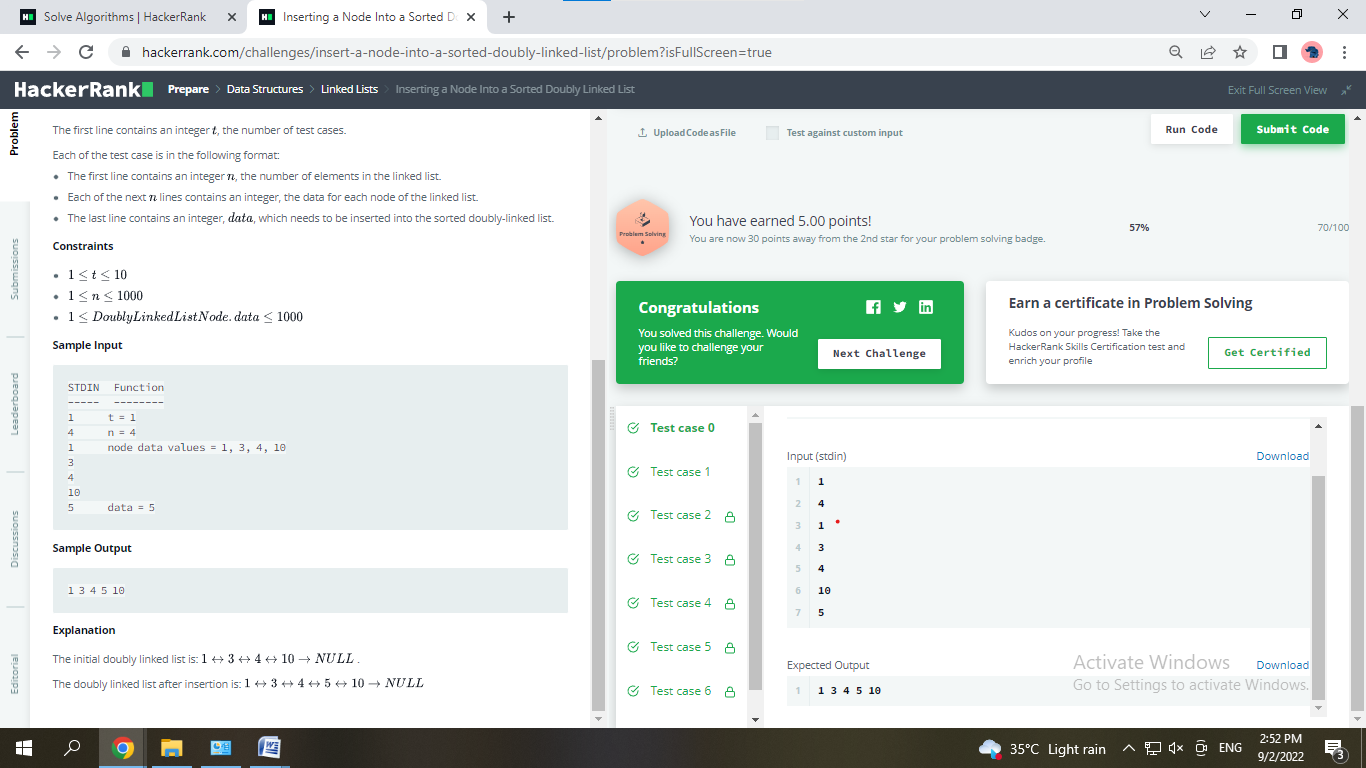
}

**4. Result/Output/Writing Summary:**









**Experiment 3.2**

1. **Aim/Overview of the practical:**

Given the pointer to the head node of a doubly linked list, reverse the order of the nodes in place. That is, change the *next* and *prev* pointers of the nodes so that the direction of the list is reversed. Return a reference to the head node of the reversed list.

1. **Apparatus / Simulator Used:**

* Windows 7 or above
* Google Chrome

1. **Code:**

import java.io.\*;

import java.math.\*;

import java.security.\*;

import java.text.\*;

import java.util.\*;

import java.util.concurrent.\*;

import java.util.regex.\*;

public class Solution {

    static class DoublyLinkedListNode {

        public int data;

        public DoublyLinkedListNode next;

        public DoublyLinkedListNode prev;

        public DoublyLinkedListNode(int nodeData) {

            this.data = nodeData;

            this.next = null;

            this.prev = null;

        }

    }

    static class DoublyLinkedList {

        public DoublyLinkedListNode head;

        public DoublyLinkedListNode tail;

        public DoublyLinkedList() {

            this.head = null;

            this.tail = null;

        }

        public void insertNode(int nodeData) {

            DoublyLinkedListNode node = new DoublyLinkedListNode(nodeData);

            if (this.head == null) {

                this.head = node;

            } else {

                this.tail.next = node;

                node.prev = this.tail;

            }

            this.tail = node;

        }

    }

    public static void printDoublyLinkedList(DoublyLinkedListNode node, String sep, BufferedWriter bufferedWriter) throws IOException {

        while (node != null) {

            bufferedWriter.write(String.valueOf(node.data));

            node = node.next;

            if (node != null) {

                bufferedWriter.write(sep);

            }

        }

    }

static DoublyLinkedListNode reverse(DoublyLinkedListNode curr) {

    DoublyLinkedListNode temp = curr.next;

    curr.next = curr.prev;

    curr.prev = temp;

    return temp == null ? curr : reverse(temp);

}

    private static final Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) throws IOException {

        BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

        int t = scanner.nextInt();

        scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

        for (int tItr = 0; tItr < t; tItr++) {

            DoublyLinkedList llist = new DoublyLinkedList();

            int llistCount = scanner.nextInt();

            scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

            for (int i = 0; i < llistCount; i++) {

                int llistItem = scanner.nextInt();

                scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

                llist.insertNode(llistItem);

            }

            DoublyLinkedListNode llist1 = reverse(llist.head);

            printDoublyLinkedList(llist1, " ", bufferedWriter);

            bufferedWriter.newLine();

        }

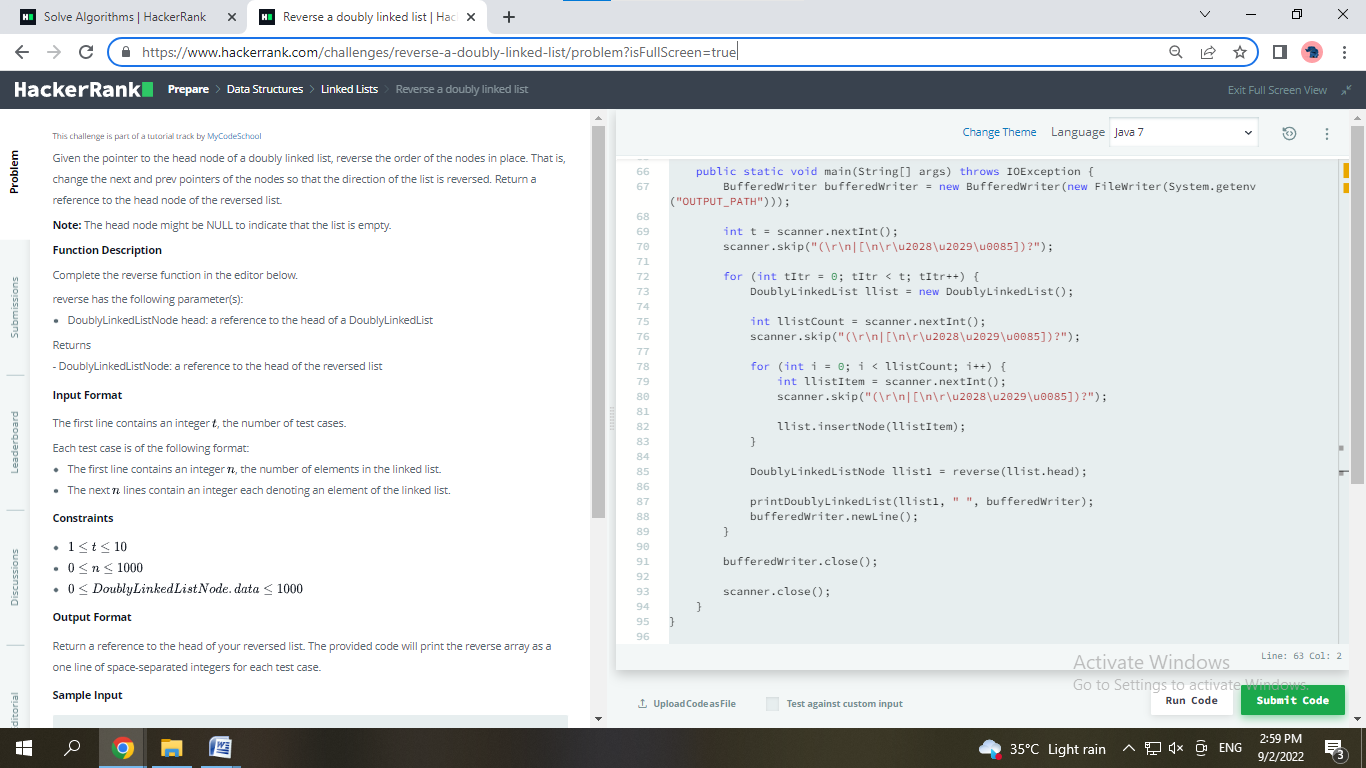
        bufferedWriter.close();

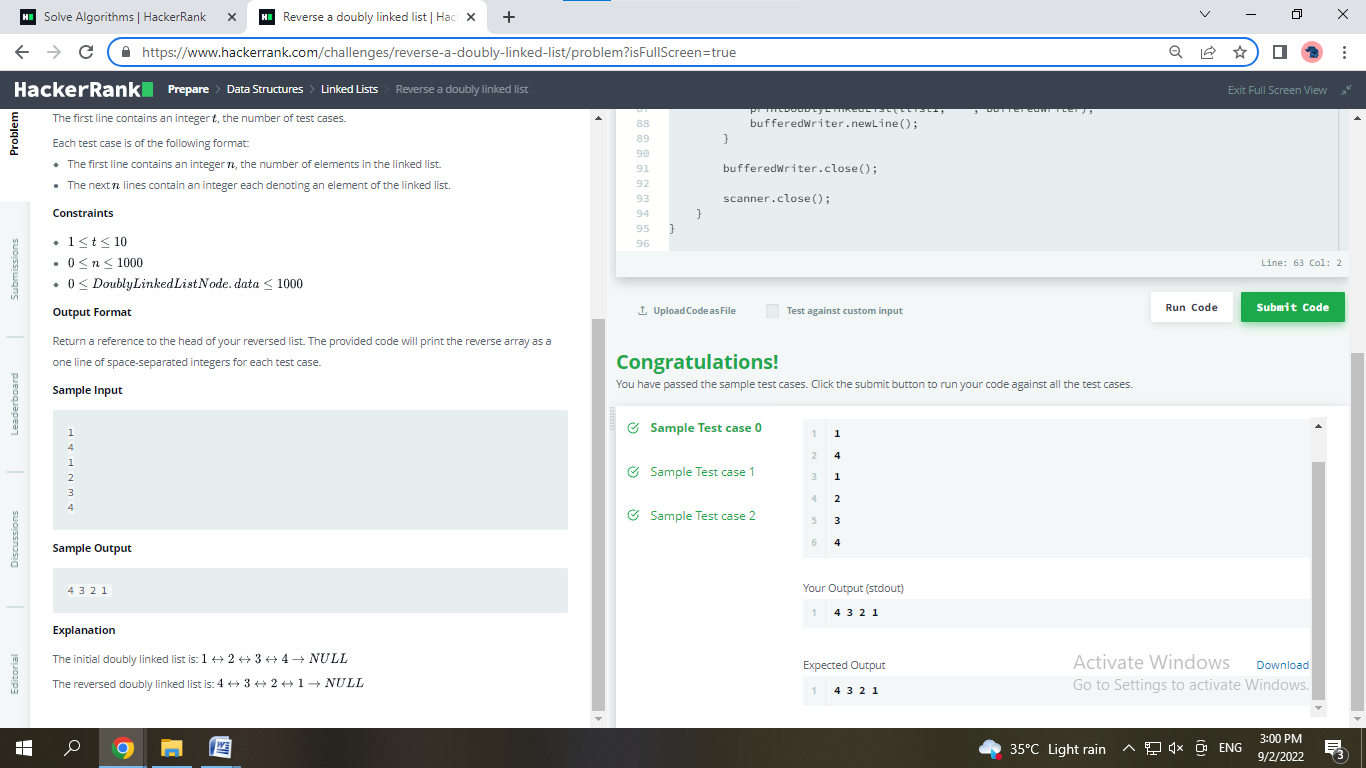
        scanner.close();

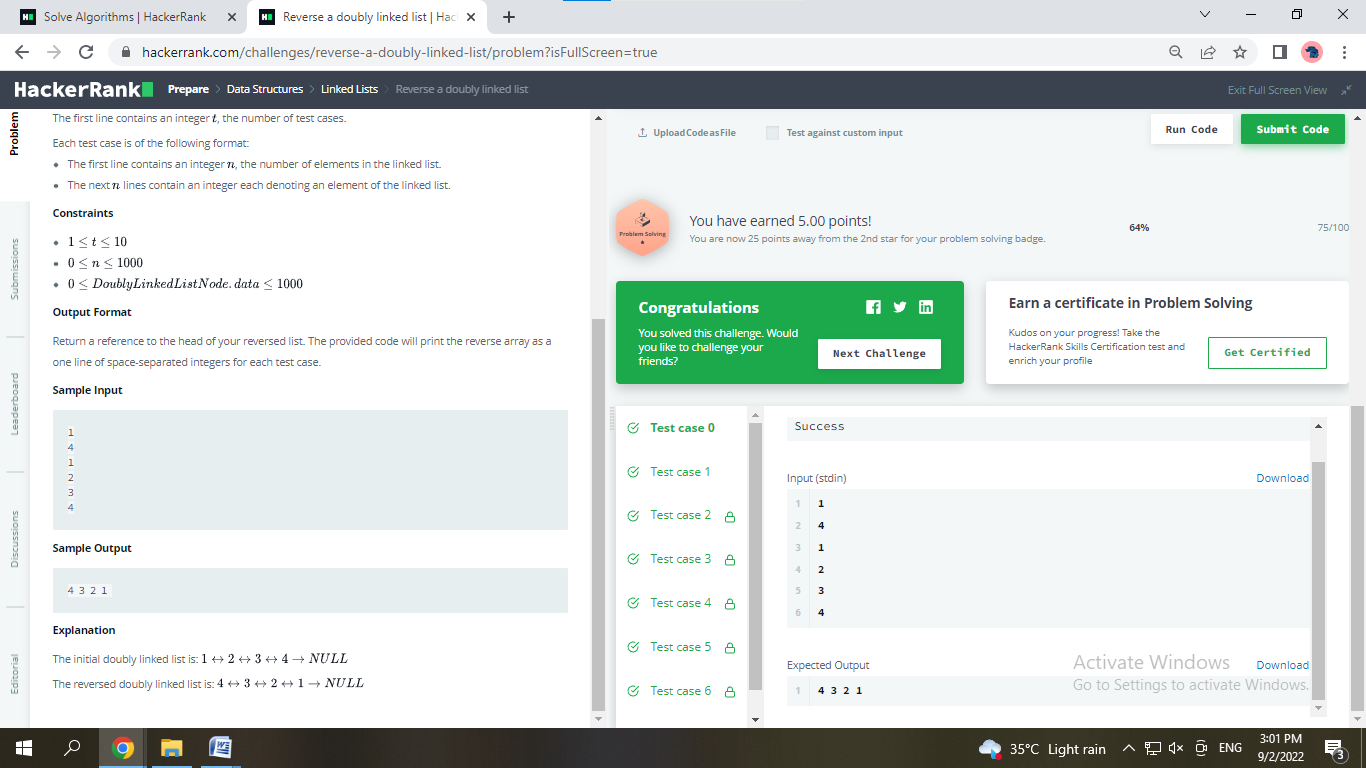
    }

}

1. **Result/Output/Writing Summary:**







**Learning outcomes (What I have learnt):**

* Learnt about Linked List data Structure
* Learn different approaches to compare two linked lists

**Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Parameters | Marks Obtained | Maximum Marks |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
|  |  |  |  |