Experiment 1.3

Student Name: Pushpraj Roy **UID: 20BCS9866**
Branch: BE-CSE **Section/Group: 617-A**
Semester: 5TH **Date of Performance: 05/08/2022**
Subject Name: CC Lab

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

PROGRAM QUESTION LINK
https://www.hackerrank.com/challenges/compare-two-linked-lists/problem?isFullScreen=true

PROGRAM CODE:

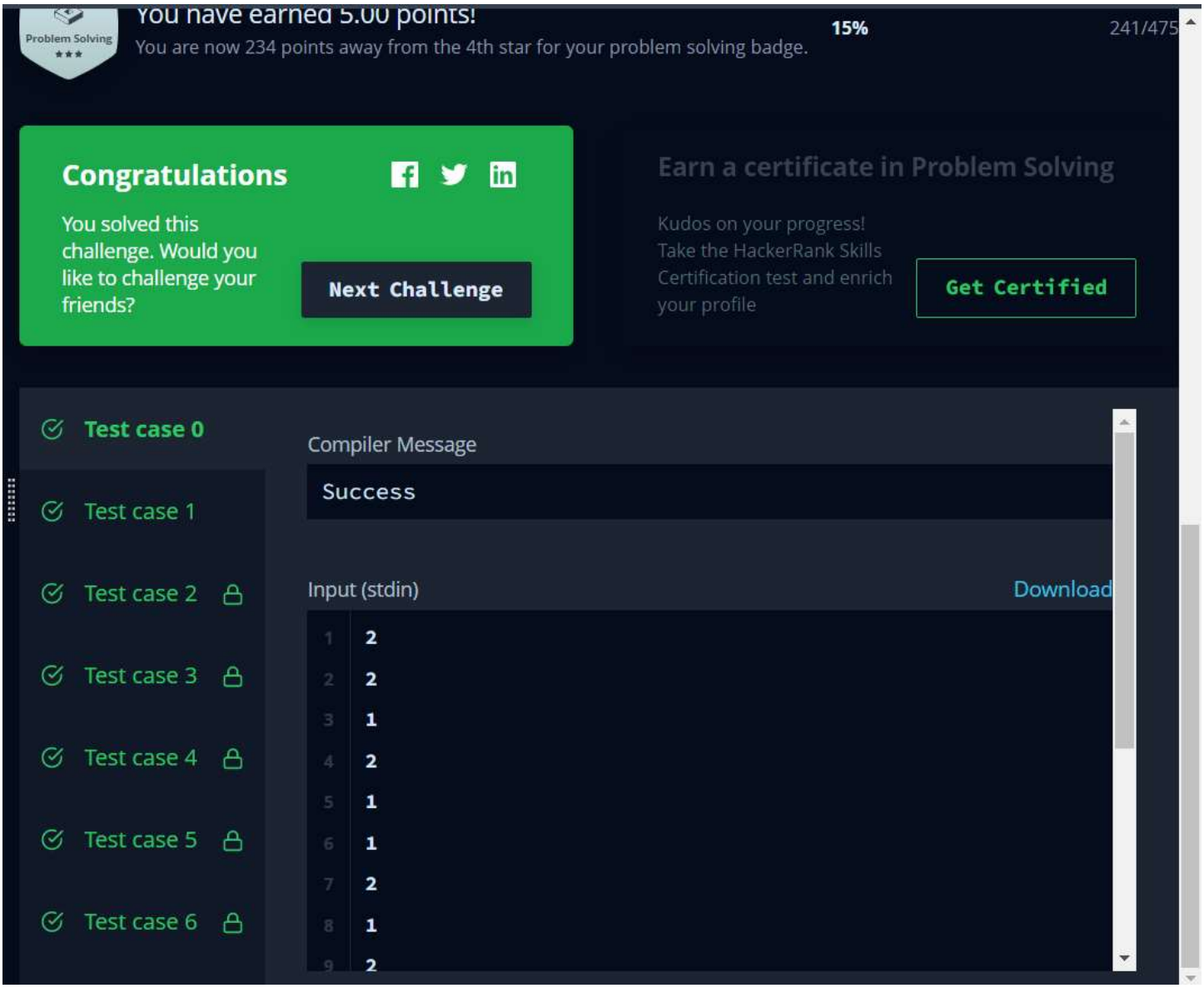
```
import java.io.*;
import java.math.*;
import java.text.*;
import java.util.*;
import java.util.regex.*;

public class Solution {

    static class SinglyLinkedListNode {
        public int data;
        public SinglyLinkedListNode next;

        public SinglyLinkedListNode(int nodeData) {
            this.data = nodeData;
            this.next = null;
        }
    }

    static class SinglyLinkedList {
        public SinglyLinkedListNode head;
        public SinglyLinkedListNode tail;
```

```
        public SinglyLinkedList() {
            this.head = null;
            this.tail = null;
        }

        public void insertNode(int nodeData) {
            SinglyLinkedListNode node = new SinglyLinkedListNode(nodeData);

            if (this.head == null) {
                this.head = node;
            } else {
                this.tail.next = node;
            }

            this.tail = node;
        }
    }

    public static void printSinglyLinkedList(SinglyLinkedListNode node, String sep, B
ufferedWriter bufferedWriter) throws IOException {
        while (node != null) {
            bufferedWriter.write(String.valueOf(node.data));

            node = node.next;

            if (node != null) {
                bufferedWriter.write(sep);
            }
        }
    }

    // Complete the compareLists function below.

    /*
     * For your reference:
     *
     * SinglyLinkedListNode {
     *     int data;
     *     SinglyLinkedListNode next;
     * }
     *
```

```
     */
    static boolean compareLists(SinglyLinkedListNode head1, SinglyLinkedListNode head
2) {
        if (head1 == null && head2 == null) {
            return true;
        } else if (head1 == null || head2 == null) {
            return false;
        } else if (head1.data != head2.data) {
            return false;
        }
        return compareLists(head1.next,head2.next);

    }



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

    public static void main(String[] args) throws IOException {
        BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.gete
nv("OUTPUT_PATH")));

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

        for (int testsItr = 0; testsItr < tests; testsItr++) {
            SinglyLinkedList llist1 = new SinglyLinkedList();

            int llist1Count = scanner.nextInt();
            scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

            for (int i = 0; i < llist1Count; i++) {
                int llist1Item = scanner.nextInt();
                scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

                llist1.insertNode(llist1Item);
            }

            SinglyLinkedList llist2 = new SinglyLinkedList();

            int llist2Count = scanner.nextInt();
            scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");
```

```
            for (int i = 0; i < llist2Count; i++) {
                int llist2Item = scanner.nextInt();
                scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

                llist2.insertNode(llist2Item);
            }

            boolean result = compareLists(llist1.head, llist2.head);

            bufferedWriter.write(String.valueOf(result ? 1 : 0));
            bufferedWriter.newLine();
        }

        bufferedWriter.close();

        scanner.close();
    }
}
```

Output :


You have earned 5.00 points!
You are now 234 points away from the 4th star for your problem solving badge.
15%
241/475
Congratulations
You solved this challenge. Would you like to challenge your friends?
Next Challenge
Earn a certificate in Problem Solving
Kudos on your progress! Take the HackerRank Skills Certification test and enrich your profile
Get Certified
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
2
2
1
2
1
1
2
1
2

PROBLEM 2 QUESTION LINK:
https://www.hackerrank.com/challenges/detect-whether-a-linked-list-contains-a-cycle/problem?isFullScreen=true

Program 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 SinglyLinkedListNode {
        public int data;
        public SinglyLinkedListNode next;

        public SinglyLinkedListNode(int nodeData) {
            this.data = nodeData;
            this.next = null;
        }
    }

    static class SinglyLinkedList {
        public SinglyLinkedListNode head;
        public SinglyLinkedListNode tail;

        public SinglyLinkedList() {
            this.head = null;
            this.tail = null;
        }

        public void insertNode(int nodeData) {
            SinglyLinkedListNode node = new SinglyLinkedListNode(nodeData);

            if (this.head == null) {
                this.head = node;
            } else {
```

```
                this.tail.next = node;
            }

            this.tail = node;
        }
    }

    public static void printSinglyLinkedList(SinglyLinkedListNode node, String sep, B
ufferedWriter bufferedWriter) throws IOException {
        while (node != null) {
            bufferedWriter.write(String.valueOf(node.data));

            node = node.next;

            if (node != null) {
                bufferedWriter.write(sep);
            }
        }
    }

    // Complete the hasCycle function below.

    /*
     * For your reference:
     *
     * SinglyLinkedListNode {
     *     int data;
     *     SinglyLinkedListNode next;
     * }
     *
     */
    static boolean hasCycle(SinglyLinkedListNode head) {

    Set<SinglyLinkedListNode> set=new HashSet<>();
    while(head!=null){
    if(set.contains(head.next))
    return true;
    set.add(head.next);
    head=head.next;
    }
    return false;
}
```

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

    public static void main(String[] args) throws IOException {
        BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.gete
nv("OUTPUT_PATH")));

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

        for (int testsItr = 0; testsItr < tests; testsItr++) {
            int index = scanner.nextInt();
            scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

            SinglyLinkedList llist = new SinglyLinkedList();

            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);
            }

            SinglyLinkedListNode extra = new SinglyLinkedListNode(-1);
            SinglyLinkedListNode temp = llist.head;

            for (int i = 0; i < llistCount; i++) {
                if (i == index) {
                    extra = temp;
                }

                if (i != llistCount-1) {
                    temp = temp.next;
                }
            }

            temp.next = extra;

            boolean result = hasCycle(llist.head);
```

```
            bufferedWriter.write(String.valueOf(result ? 1 : 0));
            bufferedWriter.newLine();
        }

        bufferedWriter.close();

        scanner.close();
    }
}
```

Output: