DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

Worksheet 2

Student Name : Anand Kashyap **UID:** 24BCS11720
Branch: BE CSE **Section/Group:** 707 (B)
Semester: 4 **Date of Performance:** 24/2/26
Subject Name: DESIGN AND ANALYSIS OF ALGORITHMS **Subject Code:** 24CSH-206

Aim:

1. To compare two linked lists and check whether they are exactly identical in length and node values.
2. To implement the Quicksort partitioning step: rearrange the array so that all elements less than the pivot come before it and all greater or equal elements come after it.

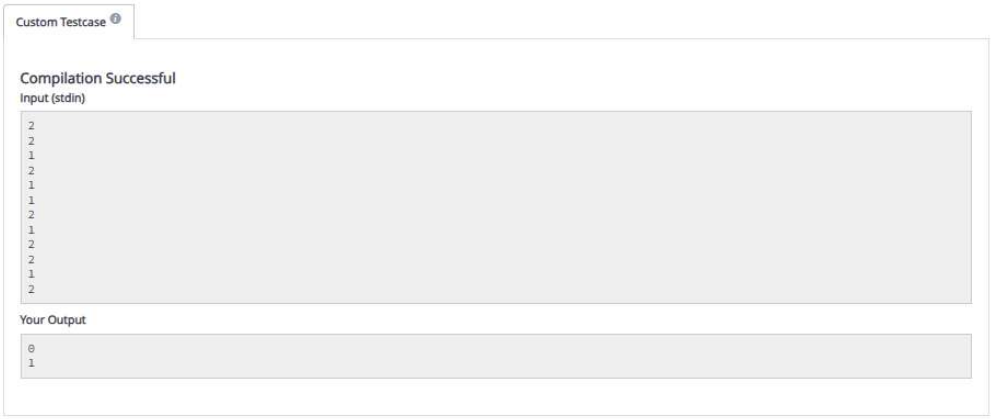
Requirements (Software): Web Browser (for accessing HackerRank), Operating System

Code:

```
// Name : Anand Kashyap
// UID  :
24BCS11720 //
Description : /*
Rearrange array elements around a pivot as part of the Quicksort algorithm. */

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,
BufferedWriter bufferedWriter) throws IOException {
    while (node != null) {
        bufferedWriter.write(String.valueOf(node.data));

        node = node.next;

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

OUTPUT:



Q – 2 To implement the Quicksort partitioning step: rearrange the array so that all elements less than the pivot come before it and all greater or equal elements come after it.

Code:

```
// Name : Anand Kashyap
// UID  : 24BCS11720
// Description :
/*
Check whether two linked lists are exactly the same in length and node values.
*/

import java.io.*;
import java.math.*;
import java.security.*;
import java.text.*;
import java.util.*;
import java.util.concurrent.*;
import java.util.regex.*;

class Result {

    /*
     * Complete the 'quickSort' function below.
     *
     * The function is expected to return an INTEGER_ARRAY.
     * The function accepts INTEGER_ARRAY arr as parameter.
     */

    public static List<Integer> quickSort(List<Integer> arr) {
```

```
 // Write your code here

    }

}

public class Solution {
    public static void main(String[] args) throws IOException {
        BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(System.in));
        BufferedWriter bufferedWriter = new BufferedWriter(new
FileWriter(System.getenv("OUTPUT_PATH")));

        int n = Integer.parseInt(bufferedReader.readLine().trim());

        String[] arrTemp = bufferedReader.readLine().replaceAll("\\s+$", "").split(" ");

        List<Integer> arr = new ArrayList<>();

        for (int i = 0; i < n; i++) {
            int arrItem = Integer.parseInt(arrTemp[i]);
            arr.add(arrItem);
        }

        List<Integer> result = Result.quickSort(arr);

        for (int i = 0; i < result.size(); i++) {
            bufferedWriter.write(String.valueOf(result.get(i)));




   if (i != result.size() - 1) {
                bufferedWriter.write(" ");
            }
        }

        bufferedWriter.newLine();

        bufferedReader.close();
        bufferedWriter.close();
    }}
```

OUTPUT:

Learning Outcomes:

After completing this experiment, the student will be able to:

1) Understand how to traverse and compare linked lists node by node.
2) Learn the concept of element comparison and equality checking in data structures.
3) Understand the partition step of Quicksort and how a pivot works.
4) Improve skills in array manipulation and algorithmic thinking.
5) Develop better understanding of time complexity and basic sorting logic.