CHANDIGARH UNIVERSITY

UNIVERSITY INSTITUTE OF NGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Submitted By:	**Submitted To:**
Vivek Kumar(21BCS8129)	Mamta Punia(E12337)
Subject Name	Competitive Coding - I
Subject Code	20CSP-314
Branch	Computer Science and Engineering
Semester	5th

Experiment No. - 3

Student Name: Vivek Kumar
UID: 21BCS8129
Branch: BE-CSE(LEET)
Section/Group: WM-20BCS-616/A
Semester: 5th
Date of Performance: 02/09/2022
Subject Name: Competitive coding - I
Subject Code: 20CSP-314

Compare two Linked list:

1. Aim/Overview of the practical:

You're given the pointer to the head nodes of two linked lists. Compare the data in the nodes of the linked lists to check if they are equal. If all data attributes are equal and the lists are the same length, return 0. Otherwise, return 1.

2. Task to be done/ Which logistics used:

Example

$llist1 = 1 \to 2 \to 3 \to NULL$
$llist2 = 1 \to 2 \to 3 \to 4 \to NULL$

The two lists have equal data attributes for the first 3 nodes. $llist2$ is longer, though, so the lists are not equal. Return 0.

Function Description

Complete the compare_lists function in the editor below.

compare_lists has the following parameters:

- SinglyLinkedListNode llist1: a reference to the head of a list
- SinglyLinkedListNode llist2: a reference to the head of a list

Returns

- int: return 1 if the lists are equal, or 0 otherwise

Input Format

The first line contains an integer t, the number of test cases.

Each of the test cases has the following format:

The first line contains an integer n, the number of nodes in the first linked list.

Each of the next n lines contains an integer, each a value for a data attribute.

The next line contains an integer m, the number of nodes in the second linked list.

Each of the next m lines contains an integer, each a value for a data attribute.

Constraints

- $1 \leq t \leq 10$
- $1 \leq n, m \leq 1000$
- $1 \leq llist1[i], llist2[i] \leq 1000$

Output Format

Compare the two linked lists and `return` 1 if the lists are equal. Otherwise, `return` 0. Do NOT print anything to stdout/console.

The output is handled by the code in the editor and it is as follows:

For each test case, in a new line, print 1 if the two lists are equal, else print 0.

Sample Input

```
2
2
1
2
1
1
2
1
2
2
1
2
```

Sample Output

```
0
1
```

Explanation

There are $t = 2$ test cases, each with a pair of linked lists.

- In the first case, linked lists are: 1 -> 2 -> NULL and 1 -> NULL
- In the second case, linked lists are: 1 -> 2 -> NULL and 1 -> 2 -> NULL

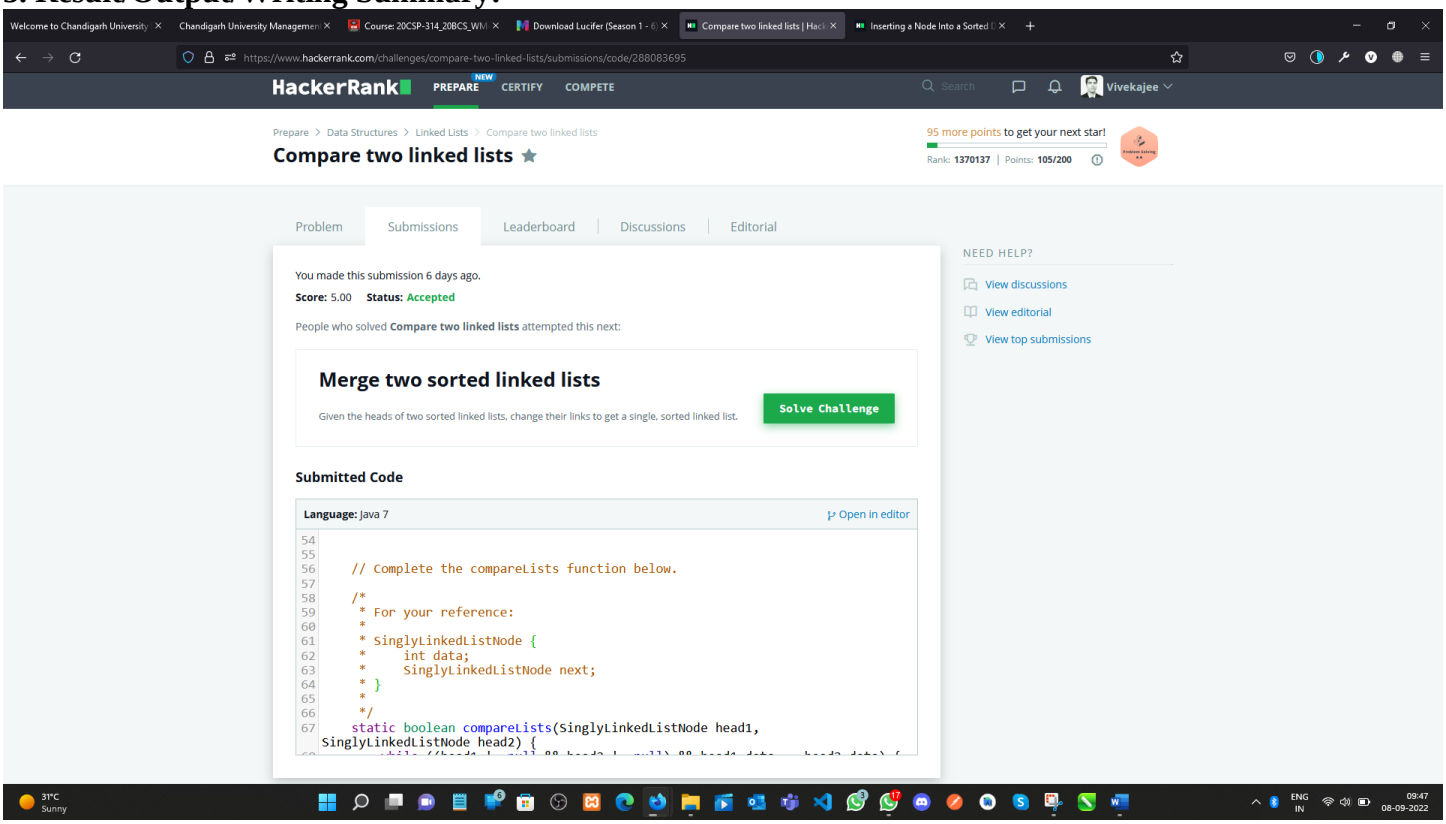
3. Hardware and Software Requirements (For programming-based labs):

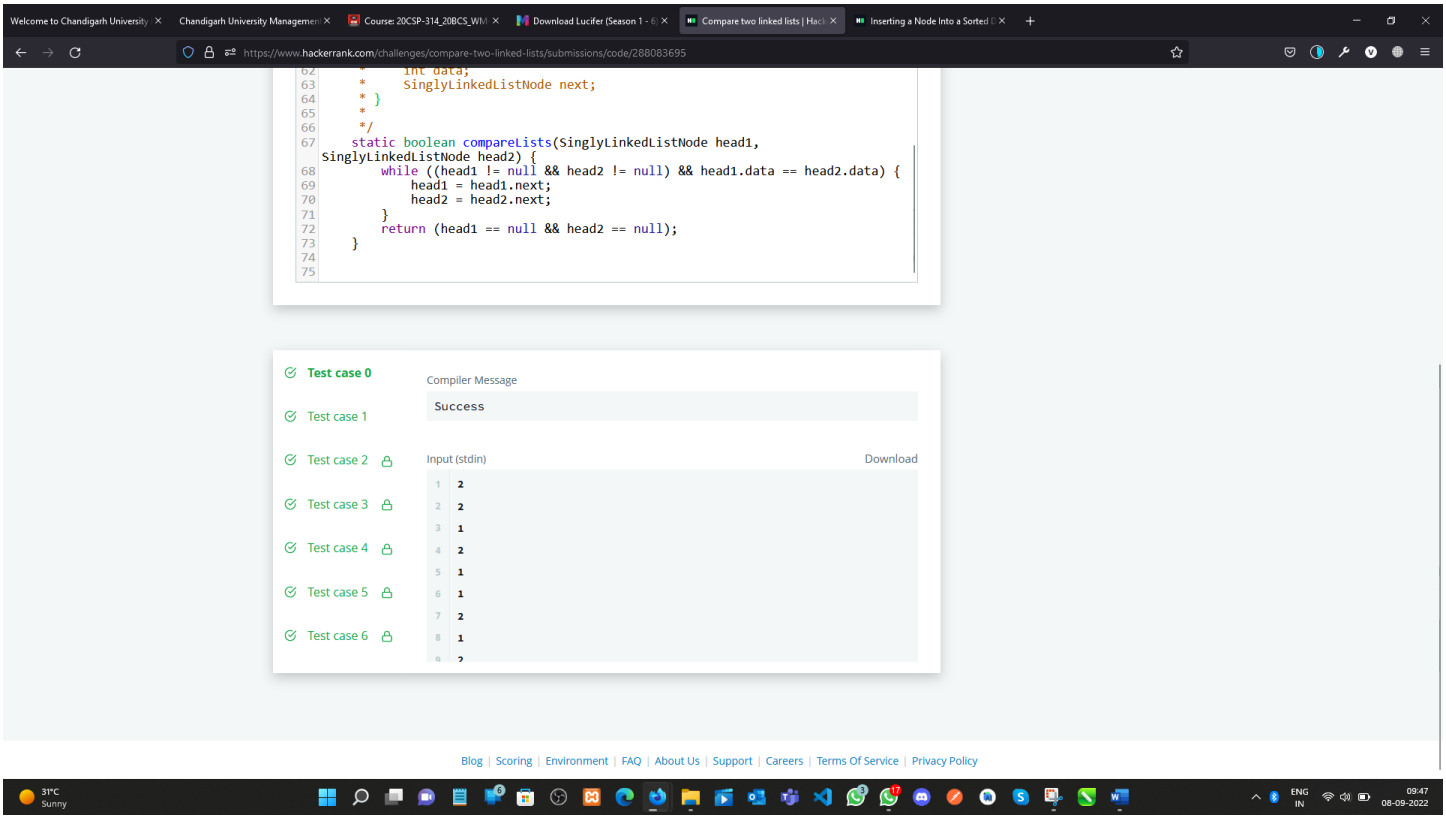
- Laptop or Desktop
- Hacker-Rank Account

4. Steps for experiment/practical/Code:

```
static boolean compareLists(SinglyLinkedListNode head1, SinglyLinkedListNode head2) {
    while ((head1 != null && head2 != null) && head1.data == head2.data) {
        head1 = head1.next;
        head2 = head2.next;
    }
    return (head1 == null && head2 == null);
}
```

5. Result/Output/Writing Summary:





Inserting a Node Into a Sorted Doubly Linked List:

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.

2. Task to be done/ Which logistics used:

Example

$head$ refers to the list $1 \leftrightarrow 2 \leftrightarrow 4 \rightarrow NULL$

$data = 3$

Return a reference to the new list: $1 \leftrightarrow 2 \leftrightarrow 3 \leftrightarrow 4 \rightarrow NULL$.

Function Description

Complete the sortedInsert function in the editor below.

sortedInsert has two parameters:

- DoublyLinkedListNode pointer head: a reference to the head of a doubly-linked list
- int data: An integer denoting the value of the $data$ field for the DoublyLinkedListNode you must insert into the list.

Returns

- DoublyLinkedListNode pointer: a reference to the head of the list

Note: Recall that an empty list (i.e., where $head =$ `NULL`) and a list with one element are sorted lists.

Input Format

The first line contains an integer t, the number of test cases.

Each of the test case is in the following format:

- The first line contains an integer n, the number of elements in the linked list.
- Each of the next n lines contains an integer, the data for each node of the linked list.
- The last line contains an integer, $data$, which needs to be inserted into the sorted doubly-linked list.

Constraints

- $1 \leq t \leq 10$
- $1 \leq n \leq 1000$
- $1 \leq DoublyLinkedListNode.data \leq 1000$

Sample Input

```
STDIN   Function
-----   --------
1       t = 1
4       n = 4
1       node data values = 1, 3, 4, 10
3
4
10
5       data = 5
```

Sample Output

```
1 3 4 5 10
```

Explanation

The initial doubly linked list is: $1 \leftrightarrow 3 \leftrightarrow 4 \leftrightarrow 10 \rightarrow NULL$.

The doubly linked list after insertion is: $1 \leftrightarrow 3 \leftrightarrow 4 \leftrightarrow 5 \leftrightarrow 10 \rightarrow NULL$

3. Hardware and Software Requirements (For programming-based labs):

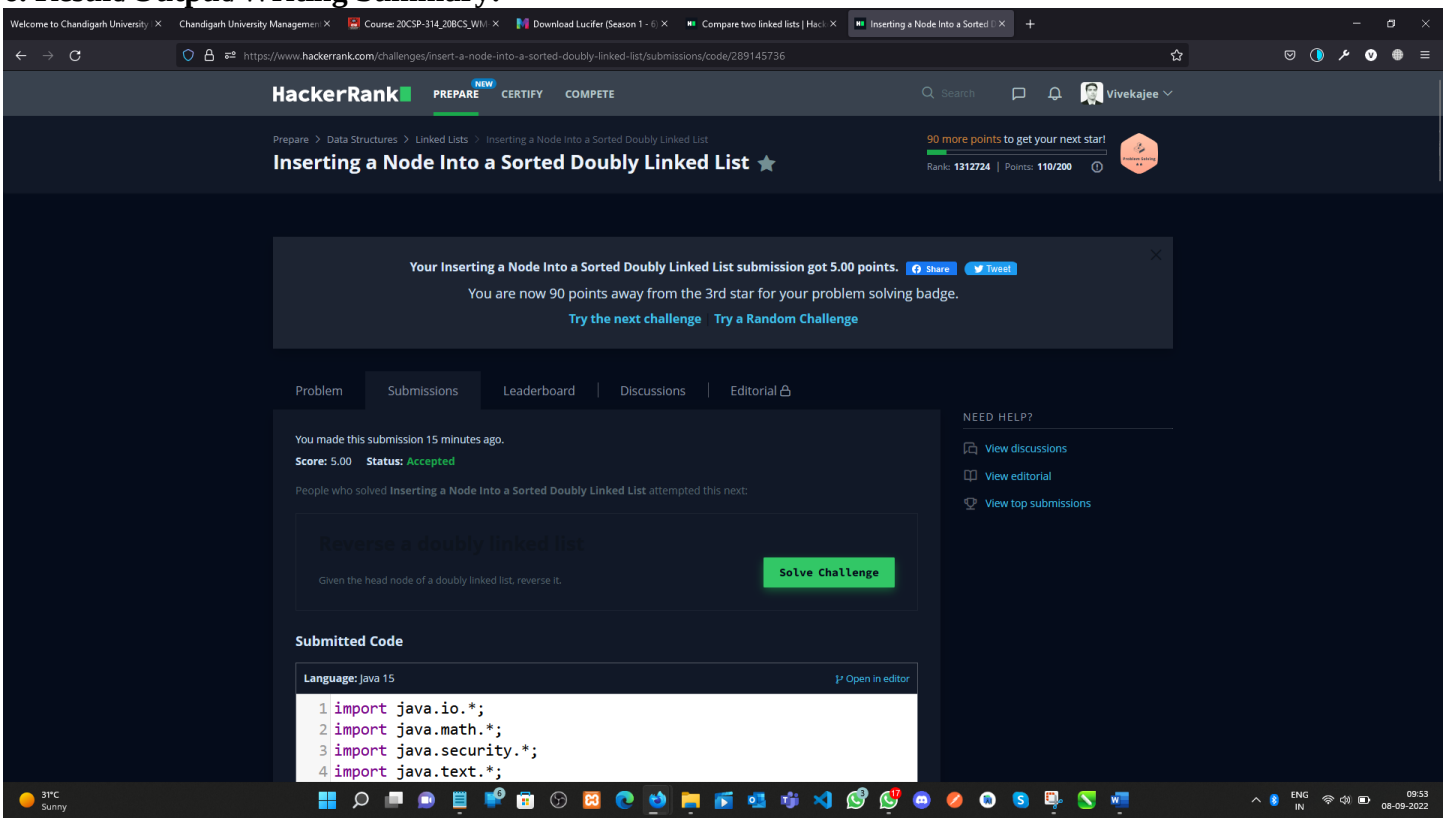
- Laptop or Desktop
- Hacker-Rank Account

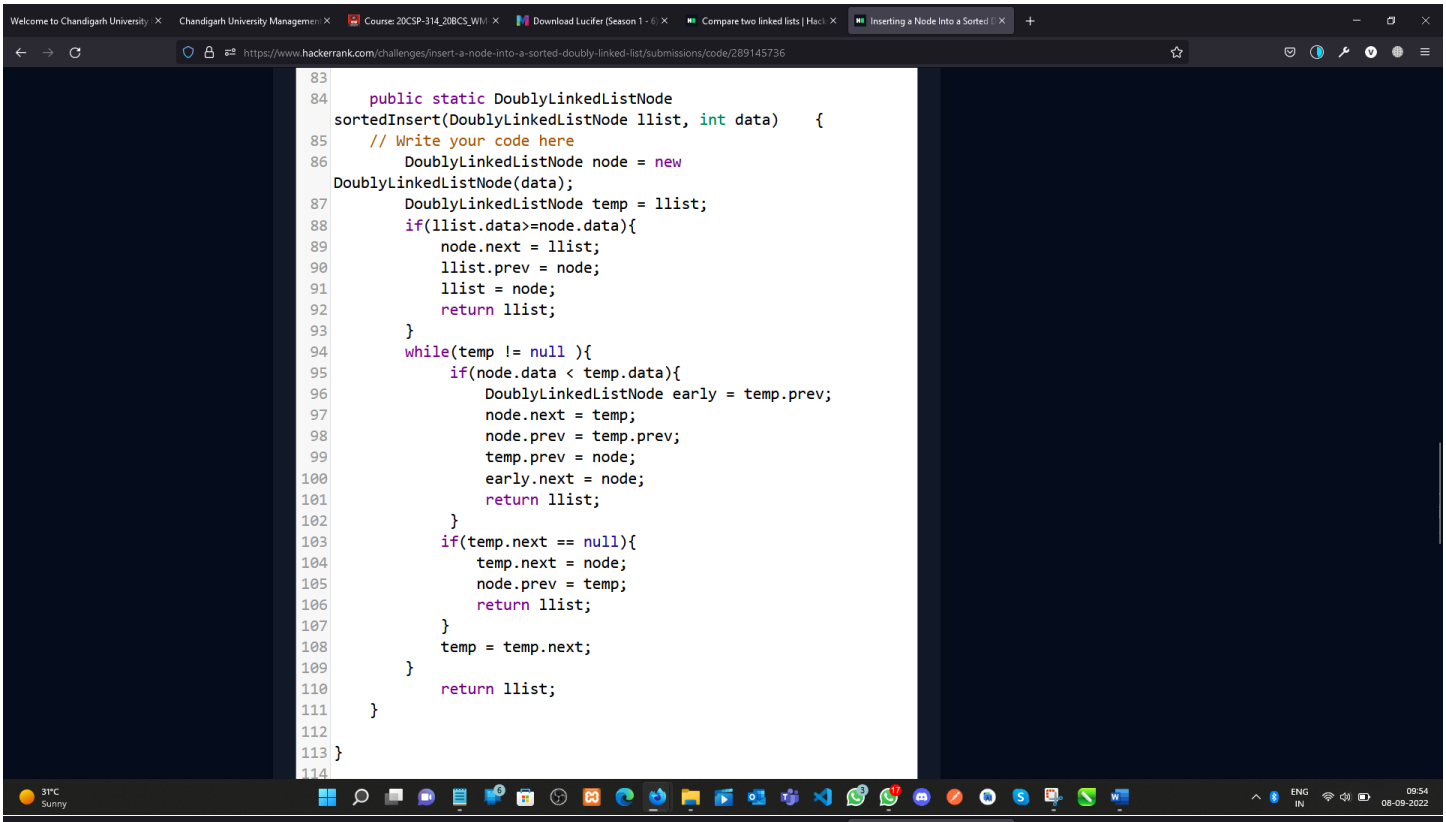
4. Steps for experiment/practical/Code:

```
public static DoublyLinkedListNode sortedInsert(DoublyLinkedListNode llist, int data)    {
   // Write your code here
      DoublyLinkedListNode node = new DoublyLinkedListNode(data);
      DoublyLinkedListNode temp = llist;
      if(llist.data>=node.data){
         node.next = llist;
         llist.prev = node;
         llist = node;
         return llist;
      }
      while(temp != null ){
          if(node.data < temp.data){
             DoublyLinkedListNode early = temp.prev;
             node.next = temp;
             node.prev = temp.prev;
             temp.prev = node;
```

```
            early.next = node;
            return llist;
         }
        if(temp.next == null){
           temp.next = node;
           node.prev = temp;
           return llist;
        }
        temp = temp.next;
    }
        return llist;
  }
```

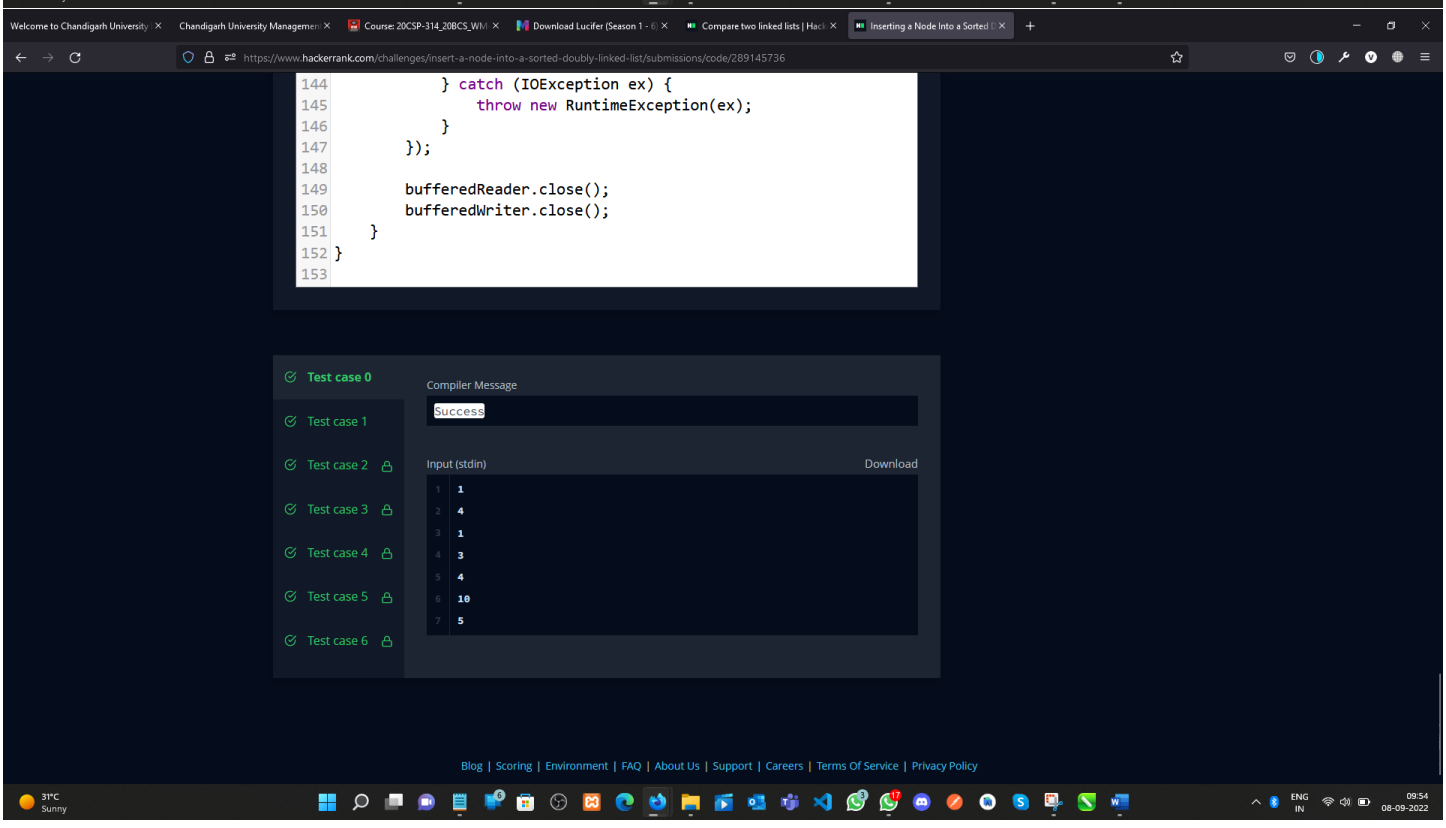
6. Result/Output/Writing Summary:





```
    public static DoublyLinkedListNode sortedInsert(DoublyLinkedListNode llist, int data)    {
    // Write your code here
        DoublyLinkedListNode node = new DoublyLinkedListNode(data);
        DoublyLinkedListNode temp = llist;
        if(llist.data>=node.data){
            node.next = llist;
            llist.prev = node;
            llist = node;
            return llist;
        }
        while(temp != null ){
             if(node.data < temp.data){
                 DoublyLinkedListNode early = temp.prev;
                 node.next = temp;
                 node.prev = temp.prev;
                 temp.prev = node;
                 early.next = node;
                 return llist;
             }
            if(temp.next == null){
                temp.next = node;
                node.prev = temp;
                return llist;
            }
            temp = temp.next;
        }
            return llist;
    }

}
```



```
            } catch (IOException ex) {
                throw new RuntimeException(ex);
            }
        });

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

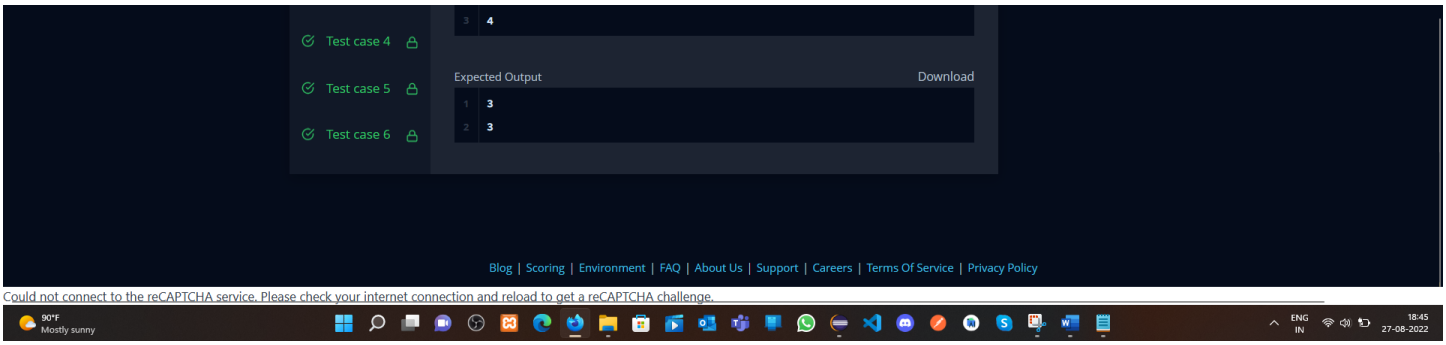
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6

Compiler Message

Success

Input (stdin) Download

```
1
4
1
3
4
10
5
```



Learning outcomes (What I have learnt):

1. Concept of LinkedList & Doubly LinkedList**.**
2. Completed my two questions.

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.			