**“Experiment 1.3”**

Student Name: **SUMIT KUMAR** UID: **20BCS8226**

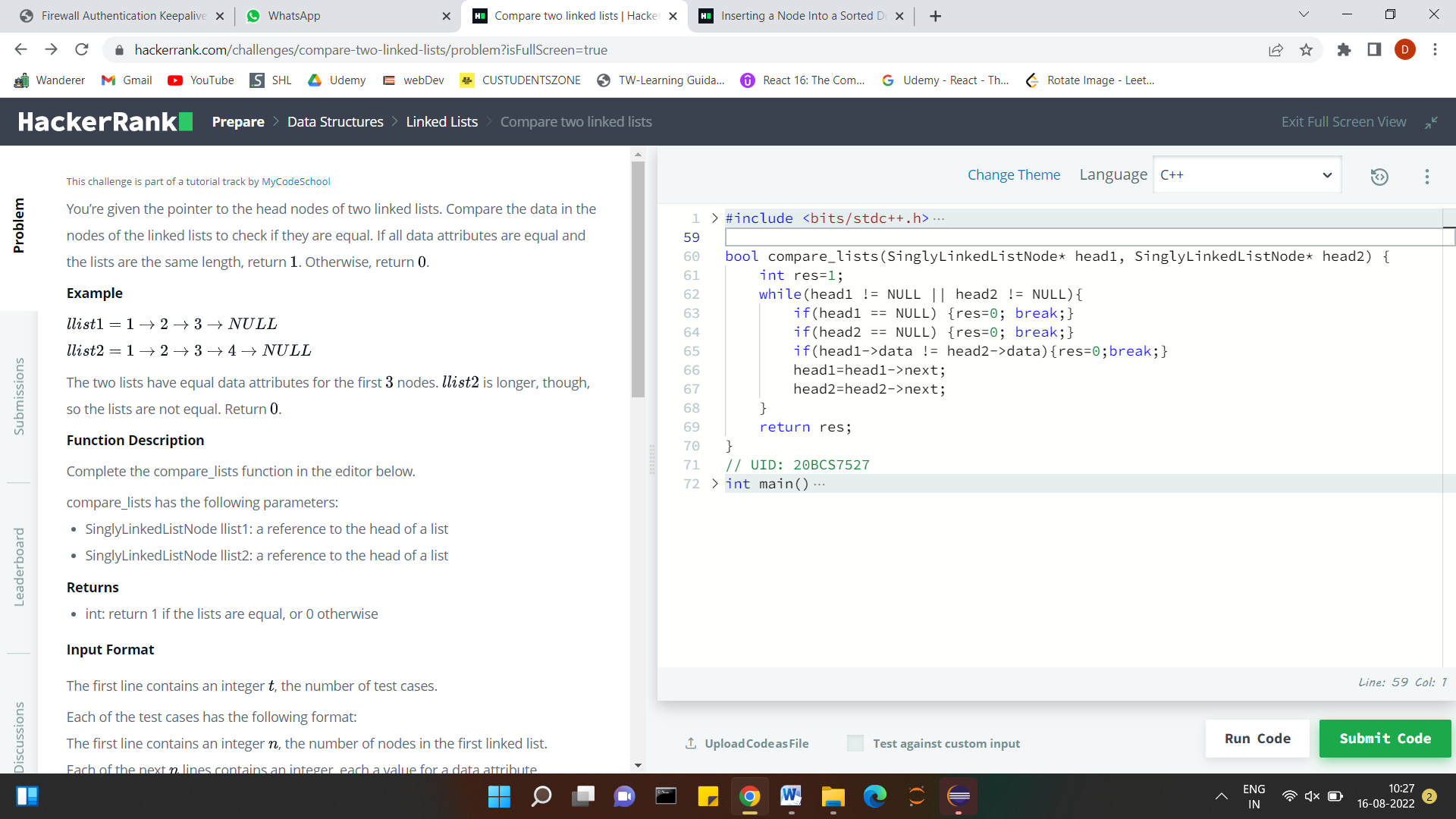
Branch: **CSE** Section/Group: **MM-808-A**

Semester: **05** Date of Submission: **05/09/22**

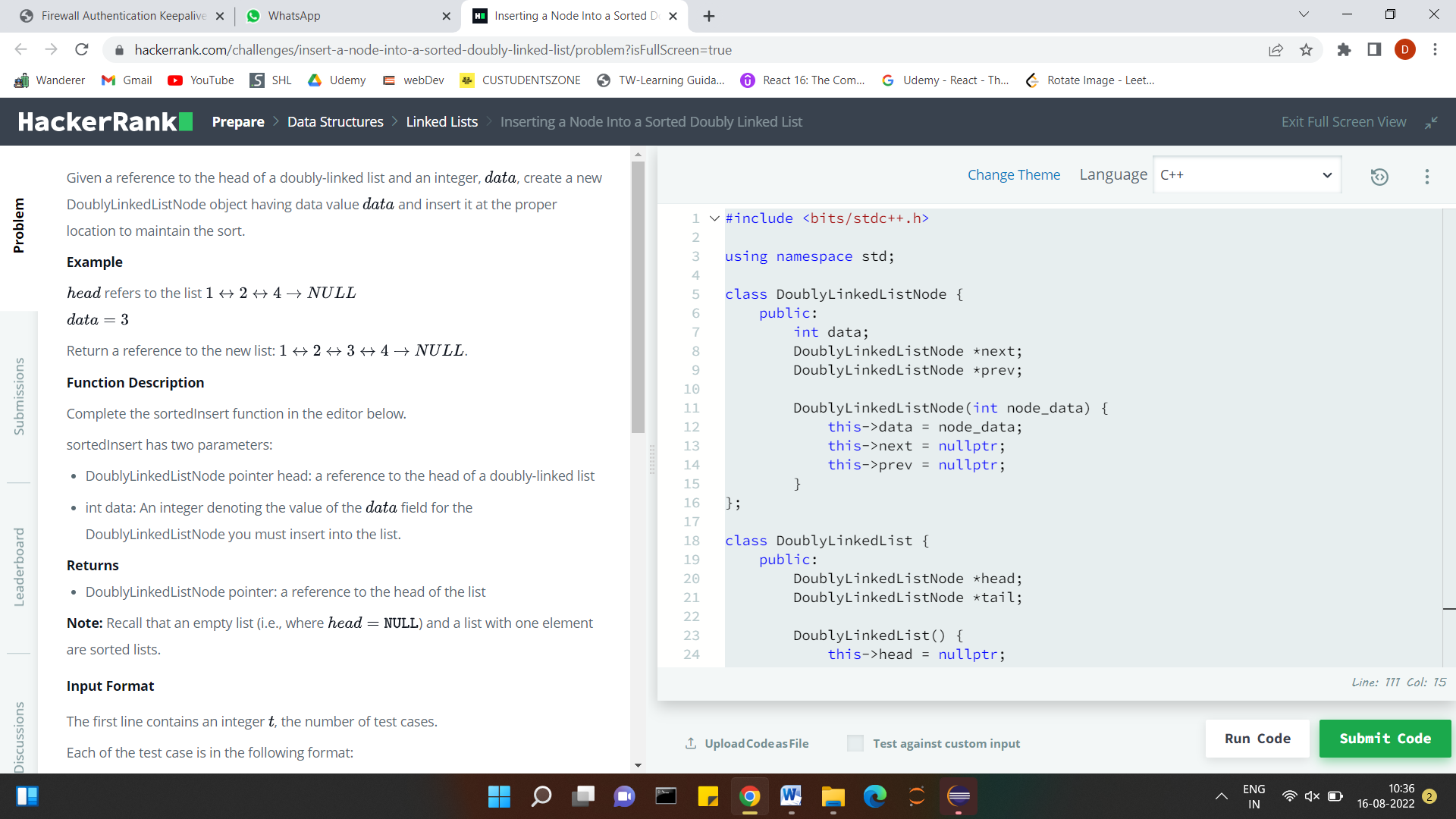
Subject Name: **Competitive Coding Lab** Subject Code: **20CSP-314**

**Aim/Overview of the practical:**

**Question 1: Compare two linked lists**



**Question 2: Inserting a Node Into a Sorted Doubly Linked List**



**Steps for experiment/practical/Code:**

**Question 1 Code:**

#include <bits/stdc++.h>

using namespace std;

class SinglyLinkedListNode {

    public:

        int data;

        SinglyLinkedListNode \*next;

        SinglyLinkedListNode(int node\_data) {

            this->data = node\_data;

            this->next = nullptr;

        }

};

class SinglyLinkedList {

    public:

        SinglyLinkedListNode \*head;

        SinglyLinkedListNode \*tail;

        SinglyLinkedList() {

            this->head = nullptr;

            this->tail = nullptr;

        }

        void insert\_node(int node\_data) {

            SinglyLinkedListNode\* node = new SinglyLinkedListNode(node\_data);

            if (!this->head) {

                this->head = node;

            } else {

                this->tail->next = node;

            }

            this->tail = node;

        }

};

void print\_singly\_linked\_list(SinglyLinkedListNode\* node, string sep, ofstream& fout) {

    while (node) {

        fout << node->data;

        node = node->next;

        if (node) {

            fout << sep;

        }

    }

}

void free\_singly\_linked\_list(SinglyLinkedListNode\* node) {

    while (node) {

        SinglyLinkedListNode\* temp = node;

        node = node->next;

        free(temp);

    }

}

bool compare\_lists(SinglyLinkedListNode\* head1, SinglyLinkedListNode\* head2) {

    int res=1;

    while(head1 != NULL || head2 != NULL){

        if(head1 == NULL) {res=0; break;}

        if(head2 == NULL) {res=0; break;}

        if(head1->data != head2->data){res=0;break;}

        head1=head1->next;

        head2=head2->next;

    }

    return res;

}

// UID: 20BCS8226

int main()

{

    ofstream fout(getenv("OUTPUT\_PATH"));

    int tests;

    cin >> tests;

    cin.ignore(numeric\_limits<streamsize>::max(), '\n');

    for (int tests\_itr = 0; tests\_itr < tests; tests\_itr++) {

        SinglyLinkedList\* llist1 = new SinglyLinkedList();

        int llist1\_count;

        cin >> llist1\_count;

        cin.ignore(numeric\_limits<streamsize>::max(), '\n');

        for (int i = 0; i < llist1\_count; i++) {

            int llist1\_item;

            cin >> llist1\_item;

            cin.ignore(numeric\_limits<streamsize>::max(), '\n');

            llist1->insert\_node(llist1\_item);

        }

        SinglyLinkedList\* llist2 = new SinglyLinkedList();

        int llist2\_count;

        cin >> llist2\_count;

        cin.ignore(numeric\_limits<streamsize>::max(), '\n');

        for (int i = 0; i < llist2\_count; i++) {

            int llist2\_item;

            cin >> llist2\_item;

            cin.ignore(numeric\_limits<streamsize>::max(), '\n');

            llist2->insert\_node(llist2\_item);

        }

        bool result = compare\_lists(llist1->head, llist2->head);

        fout << result << "\n";

    }

    fout.close();

    return 0;

}

**Question 2 Code:**

#include <bits/stdc++.h>

using namespace std;

class DoublyLinkedListNode {

    public:

        int data;

        DoublyLinkedListNode \*next;

        DoublyLinkedListNode \*prev;

        DoublyLinkedListNode(int node\_data) {

            this->data = node\_data;

            this->next = nullptr;

            this->prev = nullptr;

        }

};

class DoublyLinkedList {

    public:

        DoublyLinkedListNode \*head;

        DoublyLinkedListNode \*tail;

        DoublyLinkedList() {

            this->head = nullptr;

            this->tail = nullptr;

        }

        void insert\_node(int node\_data) {

            DoublyLinkedListNode\* node = new DoublyLinkedListNode(node\_data);

            if (!this->head) {

                this->head = node;

            } else {

                this->tail->next = node;

                node->prev = this->tail;

            }

            this->tail = node;

        }

};

void print\_doubly\_linked\_list(DoublyLinkedListNode\* node, string sep, ofstream& fout) {

    while (node) {

        fout << node->data;

        node = node->next;

        if (node) {

            fout << sep;

        }

    }

}

void free\_doubly\_linked\_list(DoublyLinkedListNode\* node) {

    while (node) {

        DoublyLinkedListNode\* temp = node;

        node = node->next;

        free(temp);

    }

}

DoublyLinkedListNode\* sortedInsert(DoublyLinkedListNode\* head, int data) {

  DoublyLinkedListNode\* node = new DoublyLinkedListNode(data);

  node->data = data;

  node->next = node->prev = NULL;

  if(head==NULL)

   return node;

 if(head->data > data){

     head->prev = node;

     node->next = head;

     return node;

 }

 DoublyLinkedListNode\* next = sortedInsert(head->next, data);

 head->next = next;

 next->prev = head;

 return head;

}

// UID: 20BCS8226

int main()

{

    ofstream fout(getenv("OUTPUT\_PATH"));

    int t;

    cin >> t;

    cin.ignore(numeric\_limits<streamsize>::max(), '\n');

    for (int t\_itr = 0; t\_itr < t; t\_itr++) {

        DoublyLinkedList\* llist = new DoublyLinkedList();

        int llist\_count;

        cin >> llist\_count;

        cin.ignore(numeric\_limits<streamsize>::max(), '\n');

        for (int i = 0; i < llist\_count; i++) {

            int llist\_item;

            cin >> llist\_item;

            cin.ignore(numeric\_limits<streamsize>::max(), '\n');

            llist->insert\_node(llist\_item);

        }

        int data;

        cin >> data;

        cin.ignore(numeric\_limits<streamsize>::max(), '\n');

        DoublyLinkedListNode\* llist1 = sortedInsert(llist->head, data);

        print\_doubly\_linked\_list(llist1, " ", fout);

        fout << "\n";

        free\_doubly\_linked\_list(llist1);

    }

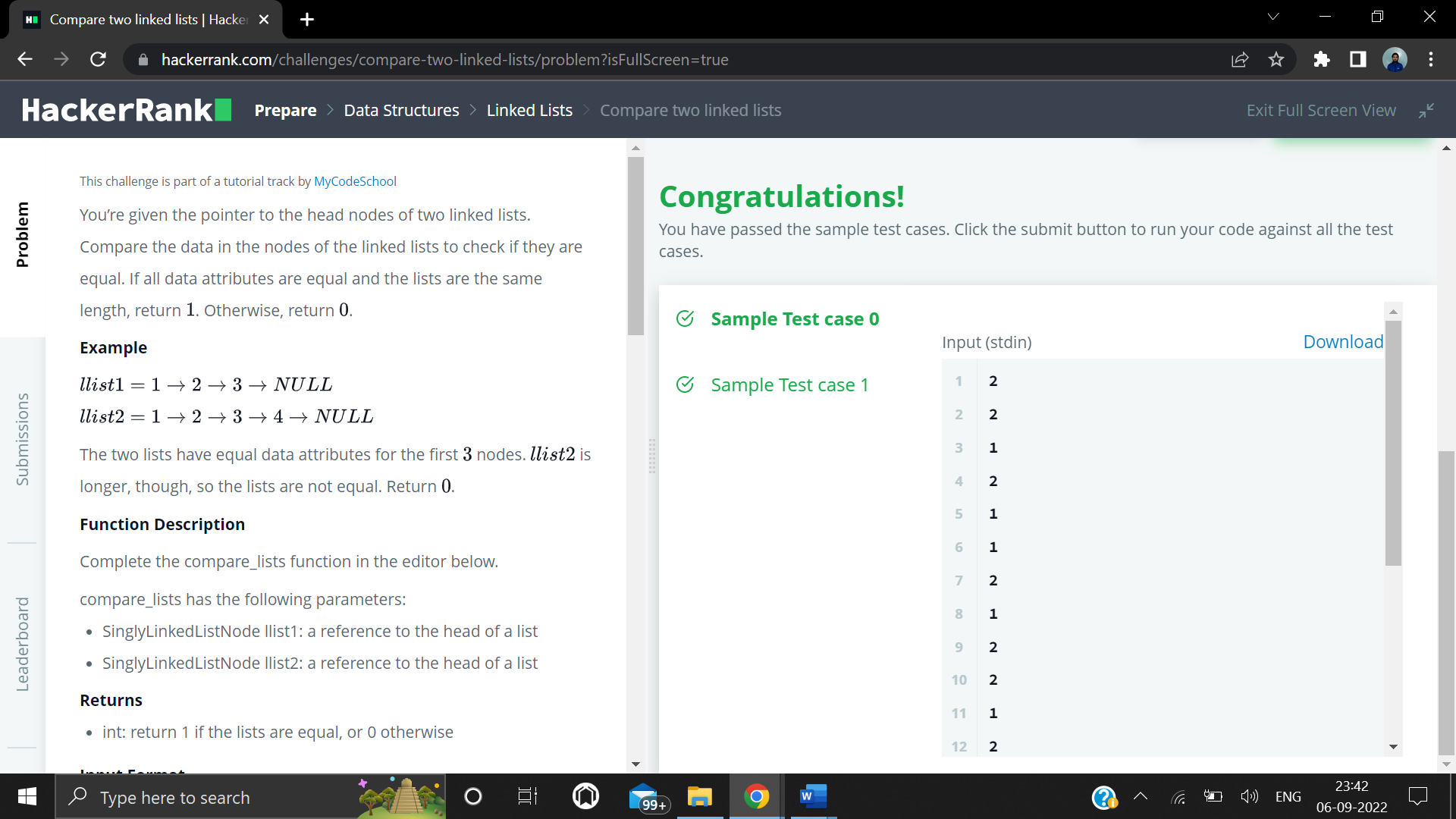
    fout.close();

    return 0;

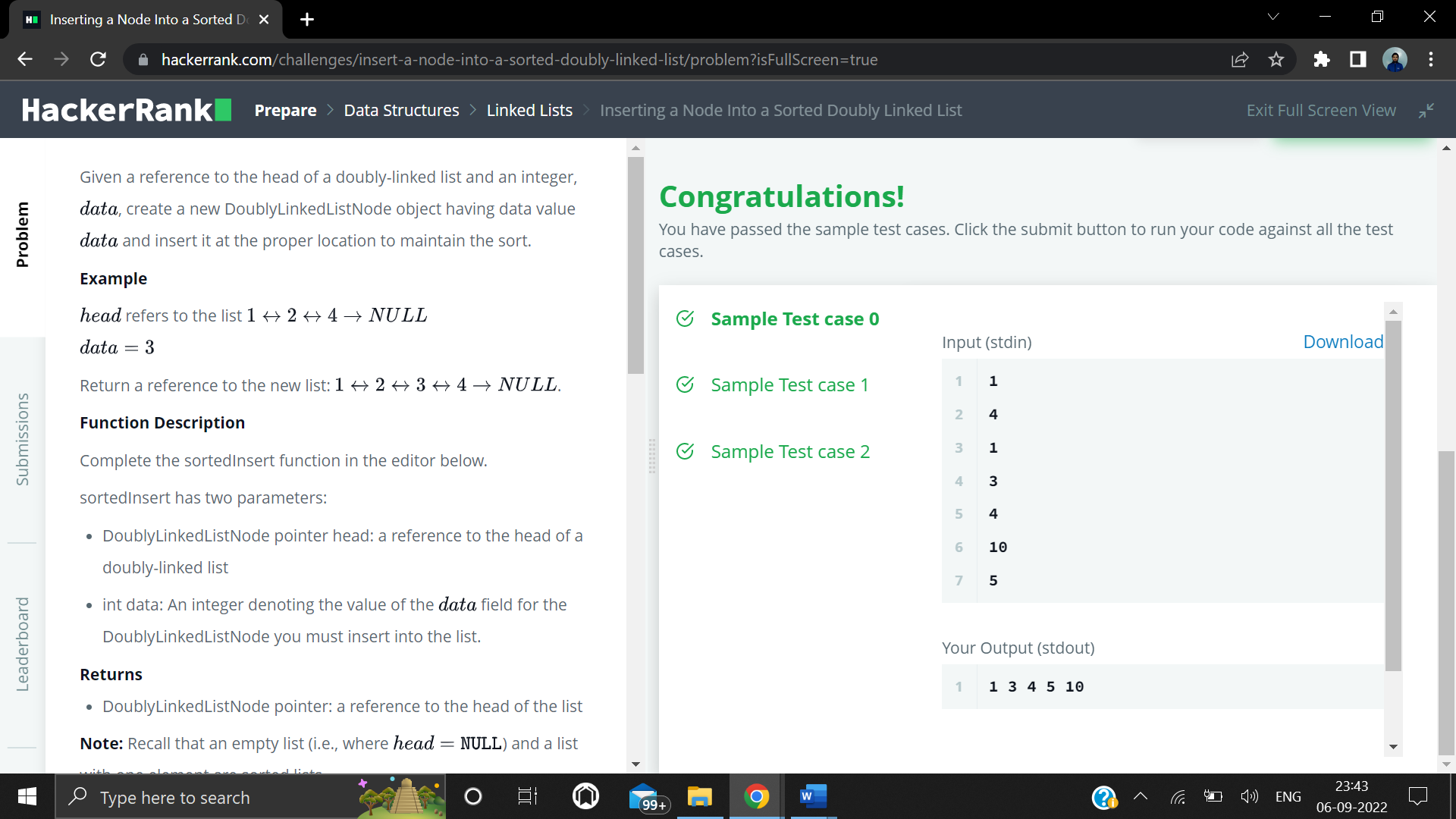
}

**Result/Output/Writing Summary:**

**Question 1 Output:**



**Question 2 Output:**



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

1. **Implementation of linked list.**

**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. |  |  |  |
|  |  |  |  |