Experiment Title 1.3

Student : ABHAY KUMAR **UID: 20BCS9222**
Branch: CSE **Section/Group:- 616-A**
Semester: 5th **D.O.P:29/08/2022**
Subject :- Competitive Coding

Aim/Overview of the practical: LINKED-LIST

Q1. 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.

Example

head refers to the list 1 <-> 2 <-> 3 <-> 4 -> NULL

data = 3

Return a reference to the new list:

1 <-> 2 <-> 3 <-> 4 -> 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 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.

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;
```

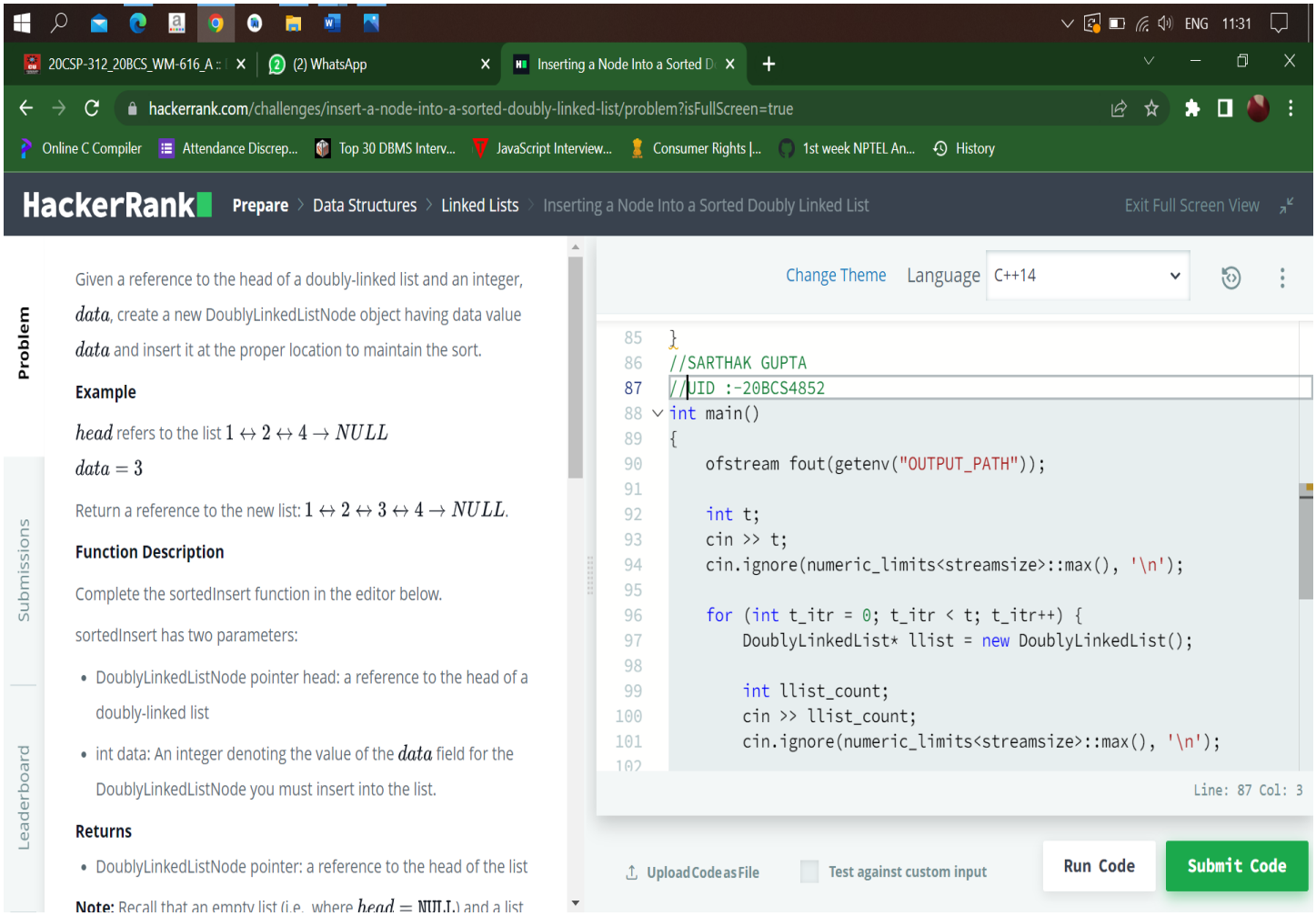
```
    } else {
    this->tail->next = node;
     node->prev = this->tail;
     }
         this->tail = node;
      }
};
void print_doubly_linked_list(DoublyLinkedListNode* node, string sep,
ofstream& fout) {
       while (node) {
       fout << node->data;
       node = node->next;
       if (node) {
          fout << sep;
       }
   }
}
void free_doubly_linked_list(DoublyLinkedListNode* node) {
   while (node) {
       DoublyLinkedListNode* temp = node;
       node = node->next;
       free(temp);
```

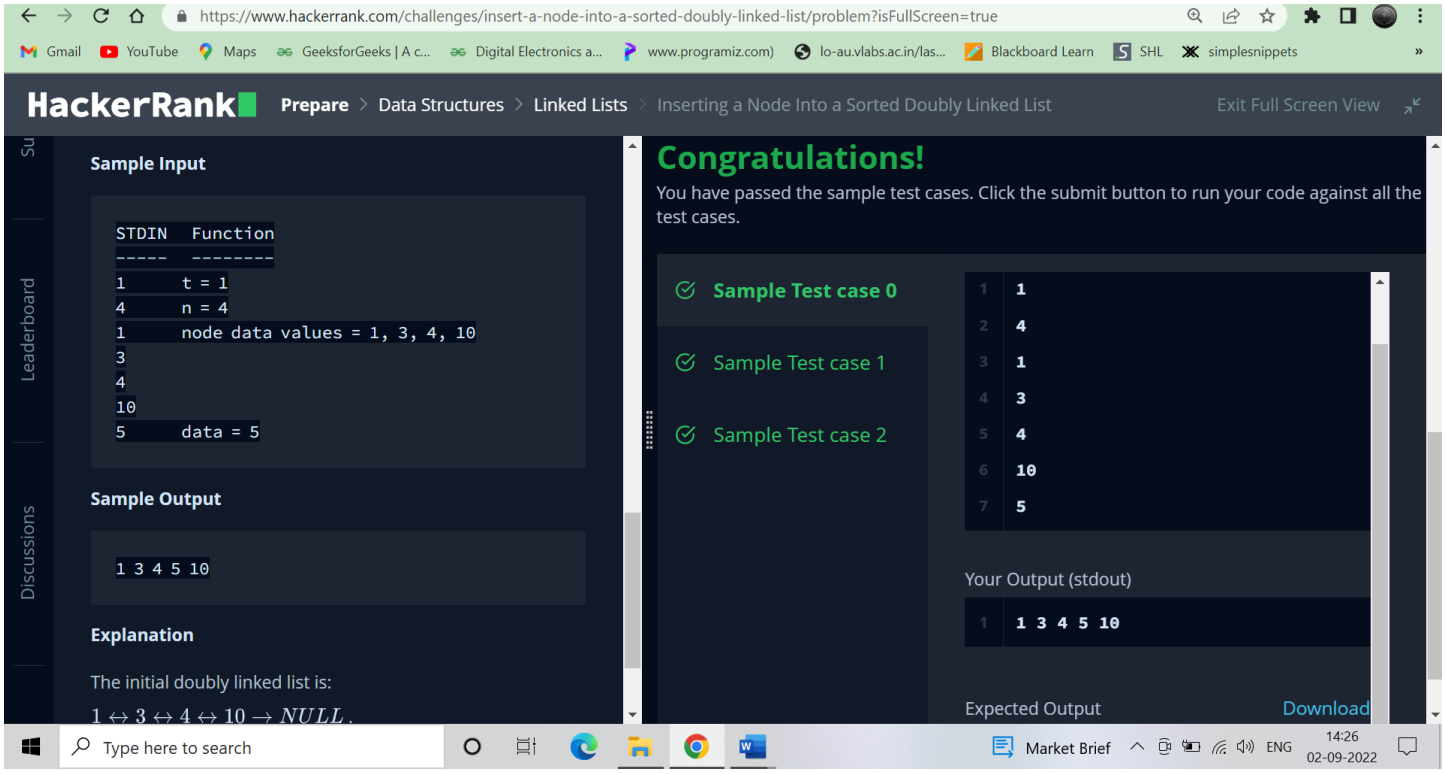
```
    }
  }
DoublyLinkedListNode* sortedInsert(DoublyLinkedListNode* head, int data) {

    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;
  }
  int main()
  {
```

```
//"SARTHAK GUPTA_20BCS4852"
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;
```

OUTPUT:-


20CSP-312_20BCS_WM-616_A
(2) WhatsApp
Inserting a Node Into a Sorted D
hackerrank.com/challenges/insert-a-node-into-a-sorted-doubly-linked-list/problem?isFullScreen=true
Online C Compiler
Attendance Discrep...
Top 30 DBMS Interv...
JavaScript Interview...
Consumer Rights |...
1st week NPTEL An...
History
HackerRank
Prepare
Data Structures
Linked Lists
Inserting a Node Into a Sorted Doubly Linked List
Exit Full Screen View
Problem
Submissions
Leaderboard
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.
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
Change Theme
Language
C++14
85 }
86 //SARTHAK GUPTA
87 //UID :-20BCS4852
88 int main()
89 {
90 ofstream fout(getenv("OUTPUT_PATH"));
91
92 int t;
93 cin >> t;
94 cin.ignore(numeric_limits<streamsize>::max(), '\n');
95
96 for (int t_itr = 0; t_itr < t; t_itr++) {
97 DoublyLinkedList* llist = new DoublyLinkedList();
98
99 int llist_count;
100 cin >> llist_count;
101 cin.ignore(numeric_limits<streamsize>::max(), '\n');
Line: 87 Col: 3
Upload Code as File
Test against custom input
Run Code
Submit Code



Q2. 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.

Note: The head node might be NULL to indicate that the list is empty.

Function Description:-

Complete the reverse function in the editor below.

reverse has the following parameter(s):

DoublyLinkedListNode head: a reference to the head of a DoublyLinkedList

Returns:-

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

Input Format :-

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

Each test case is of the following format:

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

The next n lines contain an integer each denoting an element of the linked list

CODE:-

```
#include <bits/stdc++.h>
using namespace std;

// SARTHAK GUPTA_ 20BCS4852
class DoublyLinkedListNode {
    public:
        int data;
        DoublyLinkedListNode *next;
        DoublyLinkedListNode *prev;
        DoublyLinkedListNode(int node_data) {
            this->data = node_data;
            this->next = nullptr;
            this->prev = nullptr;
        }
};
```

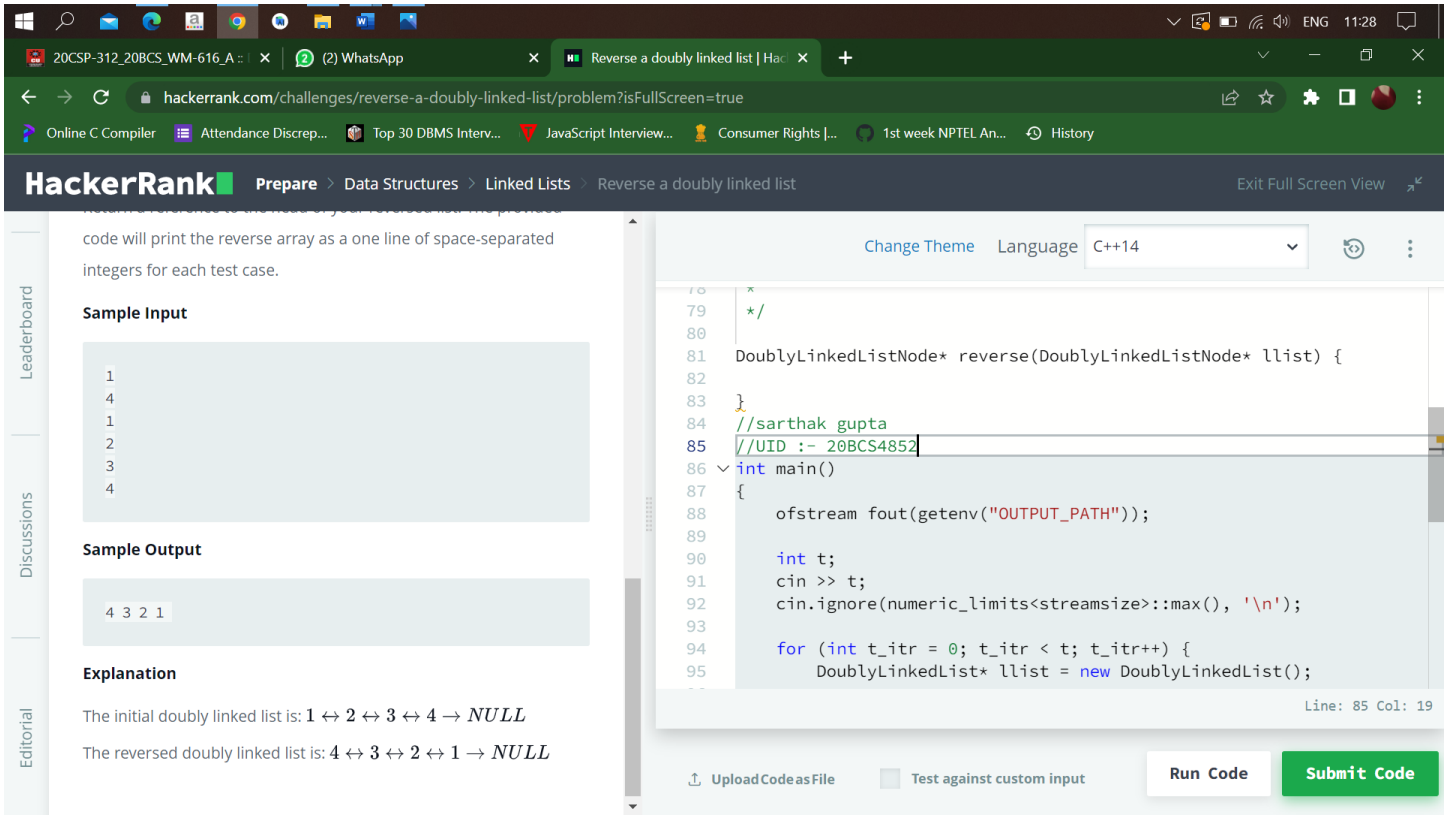
```
class DoublyLinkedList {
    public:
        DoublyLinkedListNode *head;
        DoublyLinkedListNode *tail;
        DoublyLinkedList() {
            this->head = nullptr;
            this->tail = nullptr;
        }

        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) {
```

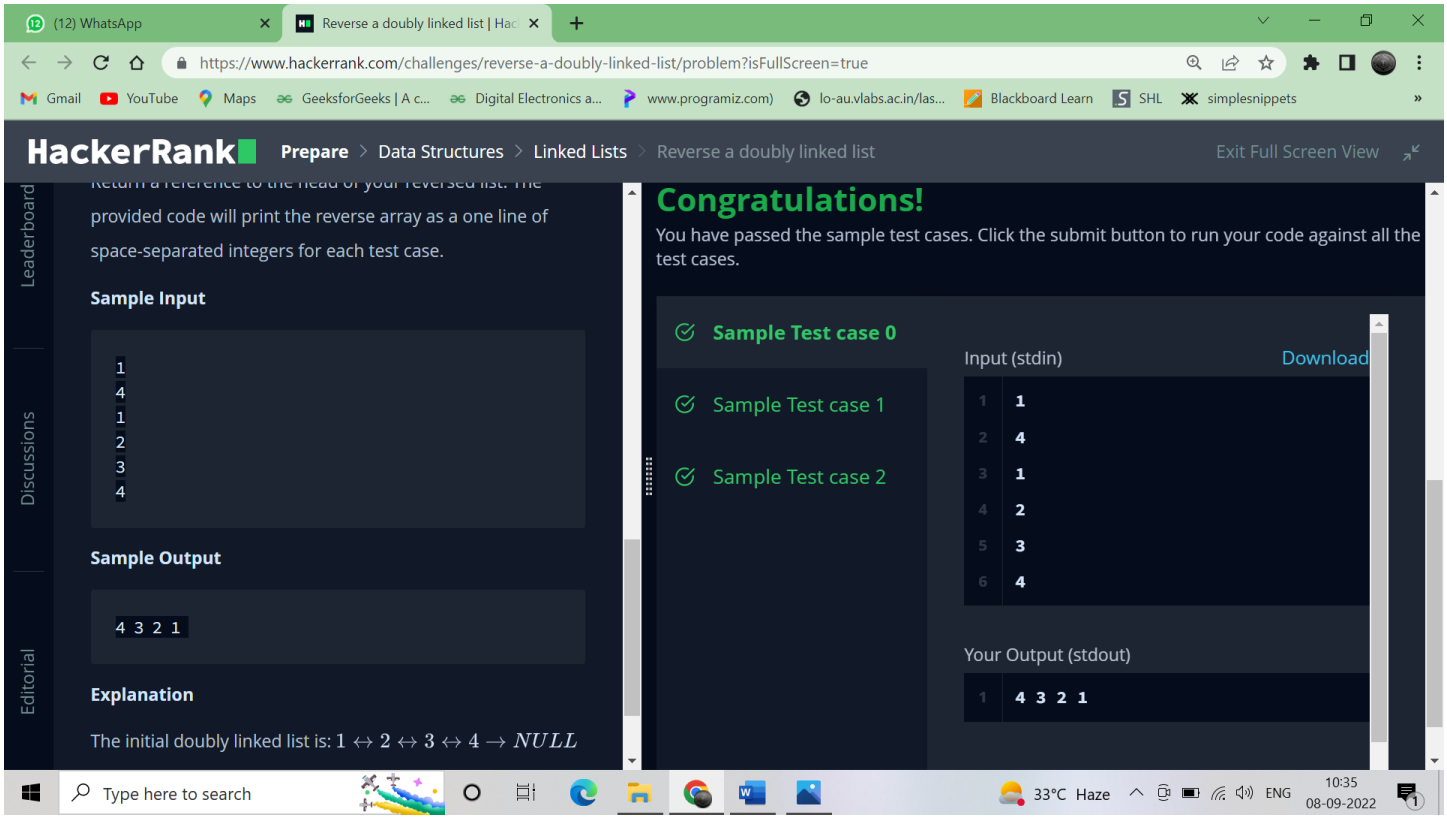
```
    while (node) {
        fout << node->data;
        node = node->next;

        if (node) {
            fout << sep;
        }
    }
}
void free_doubly_linked_list(DoublyLinkedListNode* node) {
    while (node) {
        DoublyLinkedListNode* temp = node;
        node = node->next;
        free(temp);
    }
}
// Complete the reverse function below.
/*
 * For your reference:
 * DoublyLinkedListNode {
 *     int data;
 *     DoublyLinkedListNode* next;
```

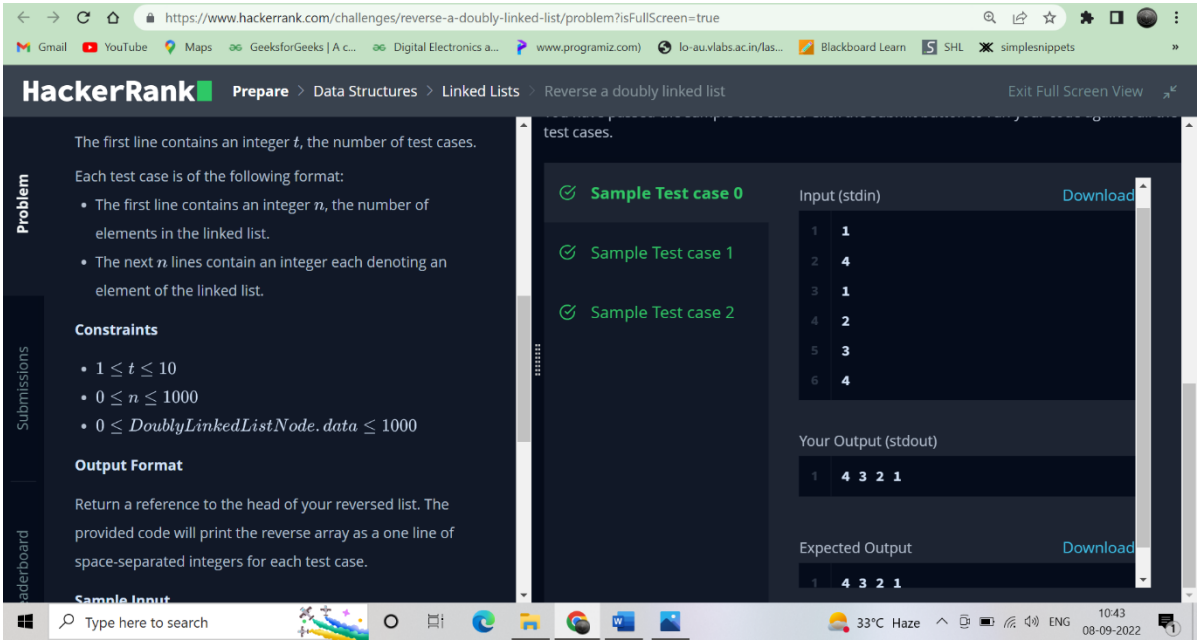
```
 *     DoublyLinkedListNode* prev;
 * };
 */
DoublyLinkedListNode* reverse(DoublyLinkedListNode* head)
{
    // Complete this function
    // Do not write the main method.
    DoublyLinkedListNode *current = head;
    DoublyLinkedListNode *temp = NULL;
    while ( current != NULL) {
        temp = current -> prev;
         current -> prev =  current -> next;
         current -> next = temp;
        current =  current -> prev;
    }
    if (temp != NULL)
        head = temp -> prev;
    return head;
}
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);
        }
        DoublyLinkedListNode* llist1 = reverse(llist->head);
        print_doubly_linked_list(llist1, " ", fout);
        fout << "\n";
        free_doubly_linked_list(llist1);
    }
    fout.close();
    return 0;
}
```

Result/Output:







Learning Outcomes:

1) Learn about C++.
2) Learn about LinkedList and how to implement it
3) Learn About DoublyLinkedList.