**Worksheet-1.3**

**Student Name:-** Pushpraj Roy **UID:-** 20BCS9866

**Branch:-** BE- CSE **Section/Group:-** WM\_617 “A”

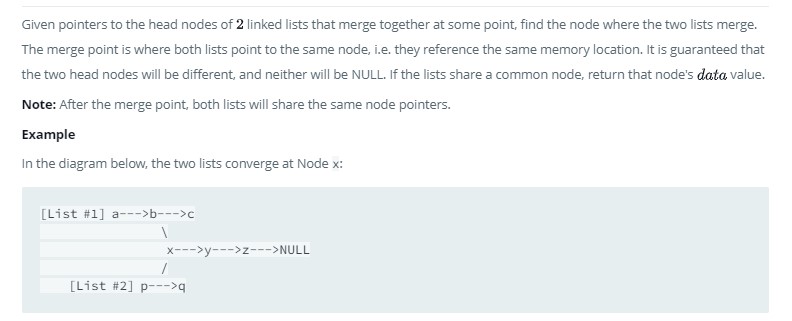
**Subjetct Code:-** 20CSP-314 **Semester:-** 5th

**Subject Name:-** Competitive Coding Lab

# Question 1-

1. **Aim:** Find Merge point of two Linked Lists.

Link- [https://www.hackerrank.com/challenges/find-the-merge-point-of-two-joinedlinked-lists/problem](https://www.hackerrank.com/challenges/find-the-merge-point-of-two-joined-linked-lists/problem)



1. **Input 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);

}

}

// Complete the findMergeNode function below.

/\*

* For your reference:

\*

* SinglyLinkedListNode {
* int data;
* SinglyLinkedListNode\* next;
* };
* \*/

int findMergeNode(SinglyLinkedListNode\* head1, SinglyLinkedListNode\* head2

) {

while (head1 != NULL) {

SinglyLinkedListNode\* temp = head2; while (temp != NULL) { if(head1 == temp){

return temp->data;

}

temp = temp->next;

}

head1 = head1->next;

}

return 0;

}

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++) { int index; cin >> index;

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

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);

}

SinglyLinkedListNode\* ptr1 = llist1->head;

SinglyLinkedListNode\* ptr2 = llist2->head;

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

if (i < index) {

ptr1 = ptr1->next;

}

}

for (int i = 0; i < llist2\_count; i++) { if (i != llist2\_count-1) {

ptr2 = ptr2->next;

}

}

ptr2->next = ptr1;

int result = findMergeNode(llist1->head, llist2->head);

fout << result << "\n";

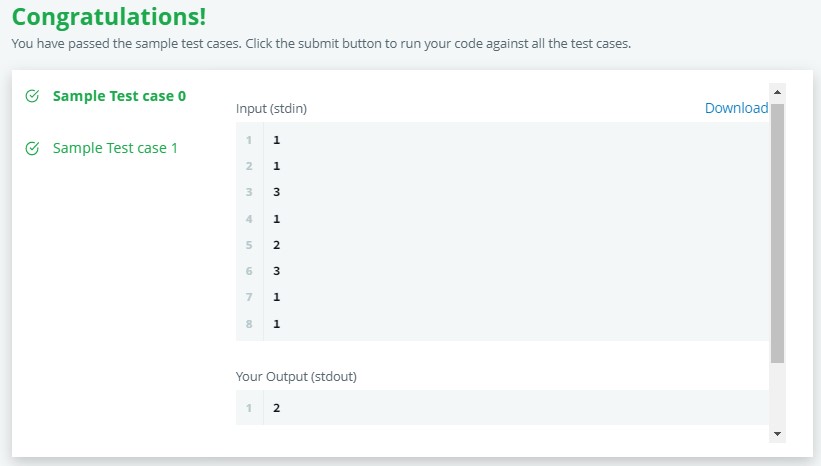
}

fout.close();

return 0;

}

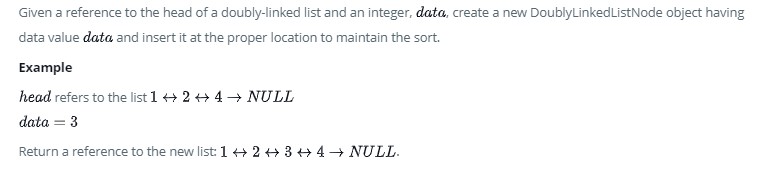
**3.Output-**



# Question 2-

1. **Aim:** Insert a node into a sorted doubly linked list.

**Link-** [**https://www.hackerrank.com/challenges/insert-a-node-into-a-sorteddoubly-linked-list/problem?isFullScreen=false**](https://www.hackerrank.com/challenges/insert-a-node-into-a-sorted-doubly-linked-list/problem?isFullScreen=false)



1. **Input 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);

}

}

/\*

* Complete the 'sortedInsert' function below.

\*

* The function is expected to return an INTEGER\_DOUBLY\_LINKED\_LIST.
* The function accepts following parameters:
* 1. INTEGER\_DOUBLY\_LINKED\_LIST llist
* 2. INTEGER data

\*/

/\*

* For your reference:

\*

* DoublyLinkedListNode {
* int data;
* DoublyLinkedListNode\* next;
* DoublyLinkedListNode\* prev;
* };

\*

\*/

DoublyLinkedListNode\* sortedInsert(DoublyLinkedListNode\* llist, int data) { DoublyLinkedListNode\* newNode = new DoublyLinkedListNode(data); if (llist == NULL){ return newNode;

}

DoublyLinkedListNode\* current = llist;

while (current->data < data && current->next != NULL) current = current->next;

if (current->data >= data)

{

if (current->prev != NULL)

{

newNode->prev = current->prev;

newNode->prev->next = newNode;

}

newNode->next = current; if (current->prev == NULL) llist = newNode;

current->prev = newNode;

} else

{

newNode->prev = current;

current->next = newNode;

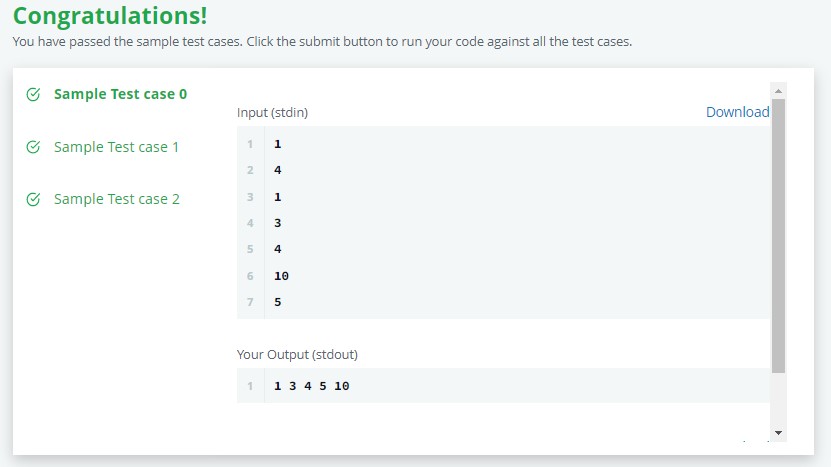
}

return llist;

}

int main()

**3.Output-**



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

**1.** We learned the concepts of implementing Linked List .

1. Here we learned how to insert a node in doubly linked List.
2. We learned how to merge linked lists and find the merge point.