Experiment1.3

Student Name: Virat Samdarshi **UID: 22BCS12648**
Branch: B.E(CSE) **Section/Group: IOT_627-B**
Semester: Fifth **Date of Performance: 01/08/24**
Subject Name: AP LAB 1 **Subject Code: 22CSP-314**

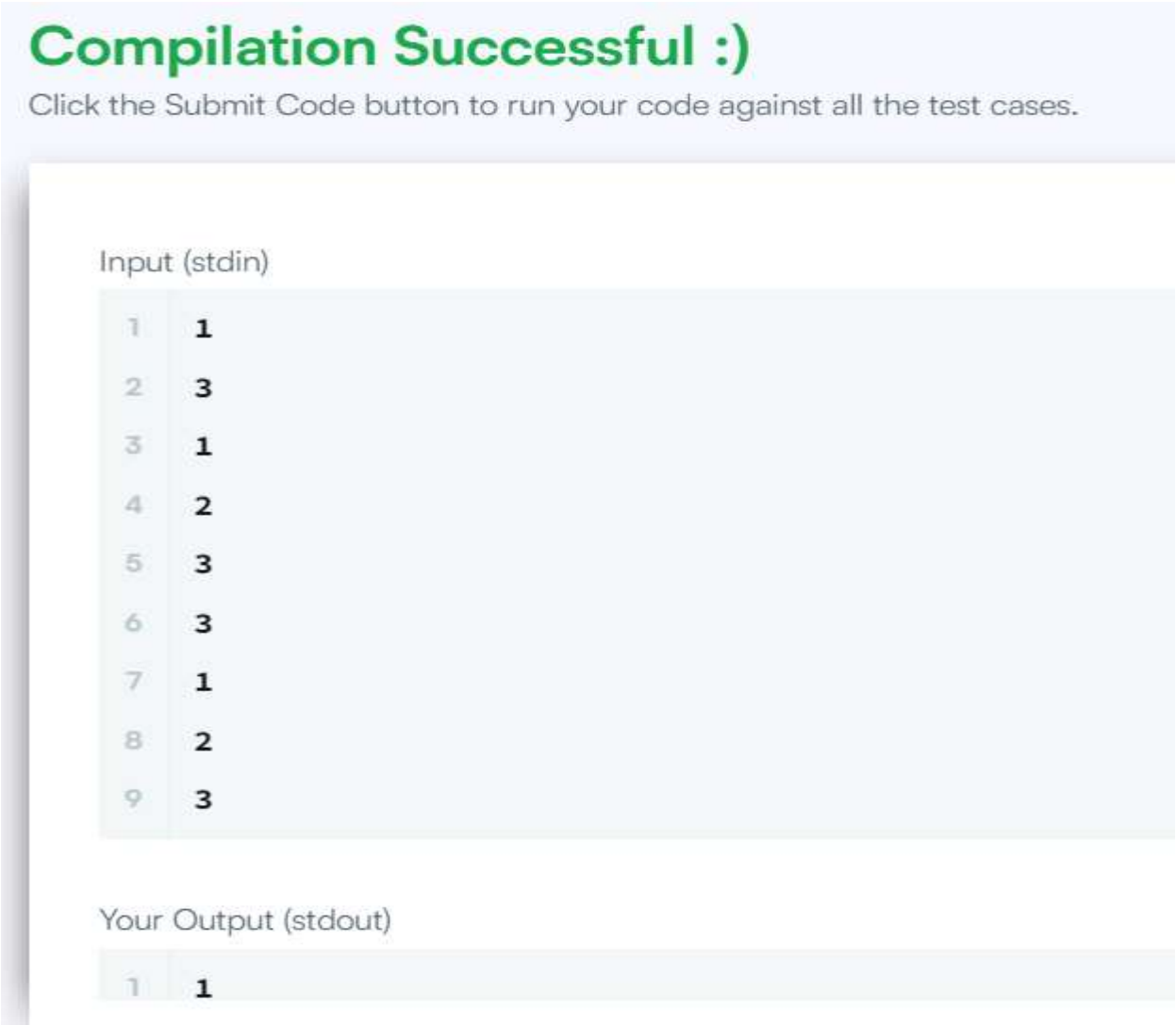
1. **Aim:** You're given the pointer to the head nodes of two linked lists. Compare the data in the nodes of the linked lists to check if they are equal. If all data attributes are equal and the lists are the same length, return 1. Otherwise, return 0.
2. **Objective:** Compare the data in the nodes of two linked lists to check if they are identical in both content and length. Return 1 if they are equal, otherwise return 0.

3. **Implementation/Code :-**

```
> #include <bits/stdc++.h>···

bool compare_lists(SinglyLinkedListNode* head1, SinglyLinkedListNode* head2) {
    while(head1!=NULL && head2!=NULL){
        if(head1->data!=head2->data){
            return 0;
        }
        else{
            head1=head1->next; head2=head2->next;
        }
}
return head1==NULL && head2==NULL ;

}
```

```
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();
```

4. Output :-



5.Time Complexity:O(n)

:

PROBLEM 2

1. **Aim:** 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.

2. **Objective:** The objective of this program is to reverse a doubly linked list in place. Given the pointer to the head node of a doubly linked list, the program will modify the next and prev pointers of each node so that the direction of the list is reversed. The program will return a reference to the new head node of the reversed list.

3. **Implementation/Code :-**

```
#include <bits/stdc++.h>···

DoublyLinkedListNode* reverse(DoublyLinkedListNode* head) {
 if (head == nullptr || head->next == nullptr) {
        return head;
    }

    DoublyLinkedListNode* prev = nullptr;
    DoublyLinkedListNode* curr = head;
    DoublyLinkedListNode* next = nullptr;

    while (curr != nullptr) {
        next = curr->next;
        curr->next = prev;
        curr->prev = next;
        prev = curr;
        curr = next;
    }

    return prev;
}
```

```
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;
```

4. Output :-

Sample Test case 0

Sample Test case 1

Sample Test case 2

Input (stdin)

```
1
4
1
2
3
4
```

Your Output (stdout)

```
4 3 2 1
```

Expected Output

```
4 3 2 1
```

5.Time Complexity: O(n).