

Experiment-3: Linked Lists

Student Name: SANSKAR AGRAWAL

Branch: CSE

Semester: 5th

UID: 20BCS5914

Section/Group: 806/B

Subject Name: CC Lab

- Aim/Overview of the practical:** Compare Two Linked Lists Problem
- Which logistics were used:** Hacker rank
- Code:**

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

4. Output:

HackerRank

Prepare > Data Structures > Linked Lists > Compare two linked lists

Exit Full Screen View

Problem

This challenge is part of a tutorial track by MyCodeSchool

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**.

Example

$l1 = 1 \rightarrow 2 \rightarrow 3 \rightarrow \text{NULL}$
 $l2 = 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow \text{NULL}$

The two lists have equal data attributes for the first **3** nodes. $l2$ 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 l1`: a reference to the head of a list
- `SinglyLinkedListNode l2`: a reference to the head of a list

Returns

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

✓ **Sample Test case 0**

9	2
10	2
11	1
12	2

✓ **Sample Test case 1**

Your Output (stdout)

1	0
2	1

Expected Output

1	0
2	1

[Download](#)

1. Aim/Overview of the practical: Cycle Detection Problem

2. Code:

```
bool has_cycle(SinglyLinkedListNode* head) {  
  
    SinglyLinkedListNode *t = head; SinglyLinkedListNode  
    *r = head;  
    if(head == NULL || head->next==NULL)  
    {  
        return false;  
    }  
    while( r!=NULL&&r->next!=NULL)  
    {  
        t = t->next;    r = r->next->next;  
        if(t==r)  
            // Condition 3  
            {  
                return true;  
            }  
        break;  
    }  
    return false;  
  
}
```

Output:

HackerRank

Prepare > Data Structures > Linked Lists > Cycle Detection

Exit Full Screen View

Problem

A linked list is said to contain a cycle if any node is visited more than once while traversing the list. Given a pointer to the head of a linked list, determine if it contains a cycle. If it does, return **1**. Otherwise, return **0**.

Example

head refers to the list of nodes **1 → 2 → 3 → NULL**

The numbers shown are the node numbers, not their data values. There is no cycle in this list so return **0**.

head refers to the list of nodes **1 → 2 → 3 → 1 → NULL**

There is a cycle where node 3 points back to node 1, so return **1**.

Function Description

Complete the `has_cycle` function in the editor below.


It has the following parameter:


- `SinglyLinkedListNode` pointer `head`: a reference to the head of the list


Returns


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	1
2	-1
3	1
4	1

Expected Output [Download](#)

1	0
---	---