Linked Lists: Detect a Cycle ☆

You have successfully solved Linked Lists: Detect a Cycle Share Tweet

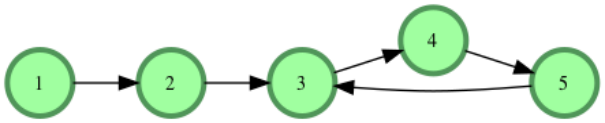
Proceed to Interview Preparation Kit

Problem Submissions Leaderboard Editorial

RATE THIS CHALLENGE

☆ ☆ ☆ ☆ ☆

A linked list is said to contain a cycle if any node is visited more than once while traversing the list. For example, in the following graph there is a cycle formed when node **5** points back to node **3**.



Function Description

Complete the function has_cycle in the editor below. It must return a boolean true if the graph contains a cycle, or false.

has_cycle has the following parameter(s):

- ***head***: a pointer to a Node object that points to the head of a linked list.

Note: If the list is empty, ***head*** will be null.

Input Format

There is no input for this challenge. A random linked list is generated at runtime and passed to your function.

Constraints

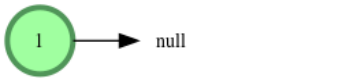
- $0 \leq$ *list size* ≤ 100

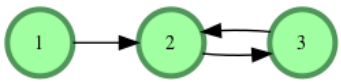
Output Format

If the list contains a cycle, your function must return true. If the list does not contain a cycle, it must return false. The binary integer corresponding to the boolean value returned by your function is printed to stdout by our hidden code checker.

Sample Input

The following linked lists are passed as arguments to your function:





Sample Output

```
0
1
```

Explanation

1. The first list has no cycle, so we return false and the hidden code checker prints **0** to stdout.
2. The second list has a cycle, so we return true and the hidden code checker prints **1** to stdout.

Current Buffer (saved locally, editable) C++

```cpp
/*
Detect a cycle in a linked list. Note that the head pointer may be 'NULL' if the list is empty.

A Node is defined as:
    struct Node {
        int data;
        struct Node* next;
    }
*/

bool has_cycle(Node* head) {
    // Complete this function
    // Do not write the main method
    Node* skip_one =head;
    Node* skip_two =head;
    //ref: https://www.youtube.com/watch?v=8ytd7HT8RnQ&ab_channel=nexTRIE
    //https://medium.com/@tuvo1106/the-tortoise-and-the-hare-floyds-algorithm-87badf5f7d41
    while(skip_two!=NULL && skip_two->next !=NULL){
        skip_one=skip_one->next;
        skip_two=skip_two->next->next;
        if(skip_one==skip_two) return true;
    }
    return false;
}
```

Line: 27 Col: 91

Upload Code as File Test against custom input Run Code Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends?

Proceed

Test case 0

Test case 1

Compiler Message

Success

Input (stdin) Download

```
1
```

Expected Output

Download

```
0
1
1
0
1
```