Cycle Detection

Problem Submissions Leaderboard Editorial

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 $\mathbf{1}$. Otherwise, return $\mathbf{0}$.

Example

$\boldsymbol{head}$ refers to the list of nodes $\mathbf{1} \rightarrow \mathbf{2} \rightarrow \mathbf{3} \rightarrow \boldsymbol{NULL}$

The numbers shown are the node numbers, not their data values. There is no cycle in this list so return $\mathbf{0}$.

$\boldsymbol{head}$ refers to the list of nodes $\mathbf{1} \rightarrow \mathbf{2} \rightarrow \mathbf{3} \rightarrow \mathbf{1} \rightarrow \boldsymbol{NULL}$

There is a cycle where node 3 points back to node 1, so return $\mathbf{1}$.

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

- int: $\mathbf{1}$ if there is a cycle or $\mathbf{0}$ if there is not

Note: If the list is empty, $\boldsymbol{head}$ will be null.

Input Format

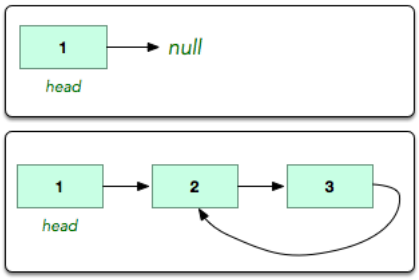
The code stub reads from stdin and passes the appropriate argument to your function. The custom test cases format will not be described for this question due to its complexity. Expand the section for the main function and review the code if you would like to figure out how to create a custom case.

Constraints

- $\mathbf{0} \leq \ \boldsymbol{list\ size}\ \leq 1000$

Sample Input

References to each of the following linked lists are passed as arguments to your function:



Sample Output

```
0
1
```

Explanation

1. The first list has no cycle, so return $\mathbf{0}$.

2. The second list has a cycle, so return **1**.

Change Theme Language Python 3

```
#!/bin/python3 ⋯

# Complete the has_cycle function below.

#
# For your reference:
#
# SinglyLinkedListNode:
#     int data
#     SinglyLinkedListNode next
#
#
def has_cycle(head):
    if head is None:
        return 0
    slow, fast = head, head
    while fast and fast.next:
        slow = slow.next
        fast = fast.next.next
        if slow == fast:
            return 1
    return 0

if __name__ == '__main__': ⋯
```

EMACS Line: 1 Col: 1

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

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Compiler Message

Success

Input (stdin) Download

```
1
-1
1
1
```

Test case 5

Expected Output

Download

```
0
```

Test case 6