

Cycle Detection

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

- *int*: **1** if there is a cycle or **0** if there is not

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

Input Format

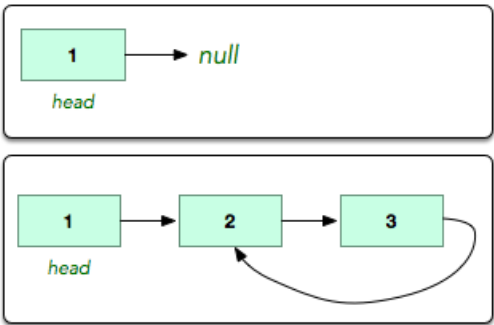
The code stub reads from stdin and passes the appropriate argument to your function. The custom test cases format will not be described for this question due to its complexity. Expand the section for the main function and review the code if you would like to figure out how to create a custom case.

Constraints

- $0 \leq \textit{list size} \leq 1000$

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 **0**.
2. The second list has a cycle, so return **1**.