Linked List Cycle II

Try to solve the Linked List Cycle II problem.

Statement

Given the head of a linked list, return the node where the cycle begins. If there is no cycle, return -1.

A cycle exists in a linked list if there is some node in the list that can be reached again by continuously following the `next` pointer. Internally, `pos` is used to denote the node's index to which the tail's next pointer is connected.

Constraints:

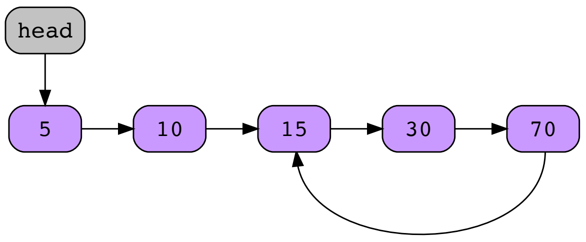
- The number of the nodes in the list is in the range $[0, 10^4]$.
- $-10^5 \leq$ `Node.val` $\leq 10^5$
- `pos` is -1 or a valid index in the linked list.

> **Note**: The `pos` parameter isn't passed as a parameter.

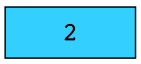
Examples

Sample example 1

Input



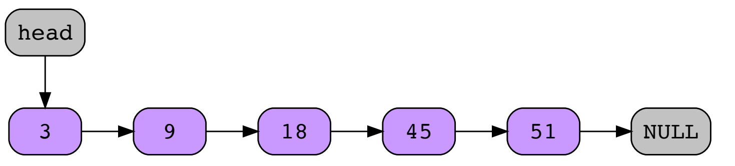
Output



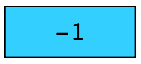
1 of 2

Sample example 2

Input



Output



2 of 2

Understand the problem

Let’s take a moment to make sure you’ve correctly understood the problem. The quiz below helps you check if you’re solving the correct problem:

Linked List Cycle II

1 What is the output if the following linked list is given as input?

Linked List = 23 → 11 → 67 → 89 → 18 → 57 → 41

pos = 3

A) 1

B) 2

C) 3

D) -1

Submit Answer

Question 1 of 3
0 attempted

Reset Quiz

Try it yourself

Implement your solution in the following coding playground:

Java

CycleDetection.java

LinkedList.java

LinkedListNode.java

```java
import java.util.*;
public class CycleDetection{
    public static LinkedListNode detectCycle(LinkedListNode head)
    // Write your code here

    return head;
  }
```

Powered by AI

Submit

Test Cases Results

Case 1 Case 2 Case 3

Input #1

[2,4,6,8,10]

Input #2

2

Linked List Cycle II

Hide Hint

You might want to go over the Fast and Slow Pointers pattern again.

Back

Top K Frequent Words

Next

Minimum Flips to Mak...

Mark as Completed