Problem 237: Delete Node in a Linked List

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

There is a singly-linked list

head

and we want to delete a node

node

in it.

You are given the node to be deleted

node

. You will

not be given access

to the first node of

head

.

All the values of the linked list are

unique

, and it is guaranteed that the given node

node

is not the last node in the linked list.

Delete the given node. Note that by deleting the node, we do not mean removing it from memory. We mean:

The value of the given node should not exist in the linked list.

The number of nodes in the linked list should decrease by one.

All the values before

node

should be in the same order.

All the values after

node

should be in the same order.

Custom testing:

For the input, you should provide the entire linked list

head

and the node to be given

node

.

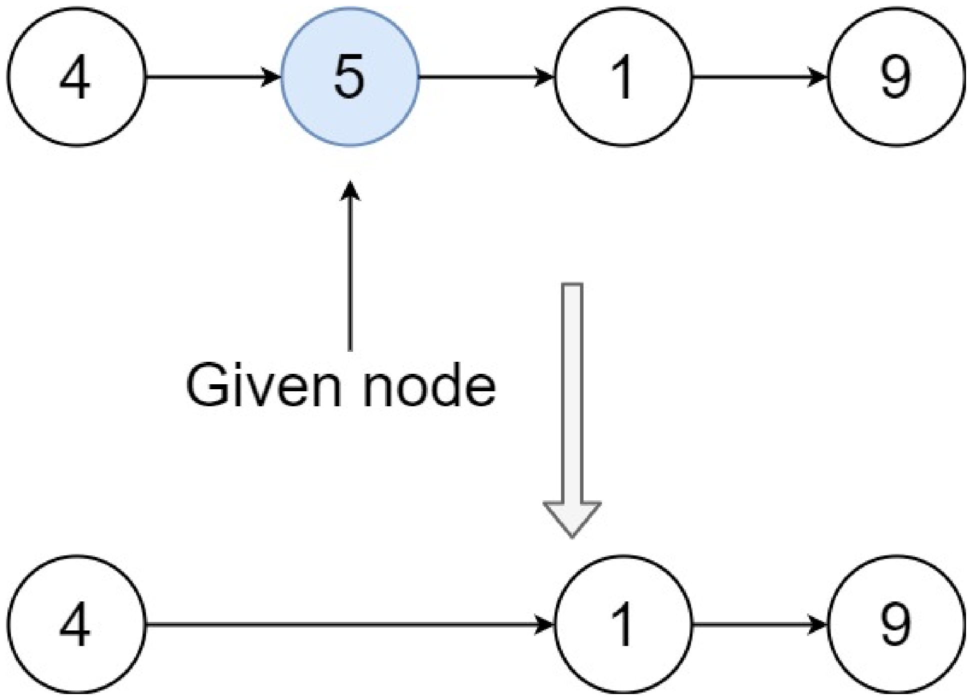
node

should not be the last node of the list and should be an actual node in the list.

We will build the linked list and pass the node to your function.

The output will be the entire list after calling your function.

Example 1:



Input:

head = [4,5,1,9], node = 5

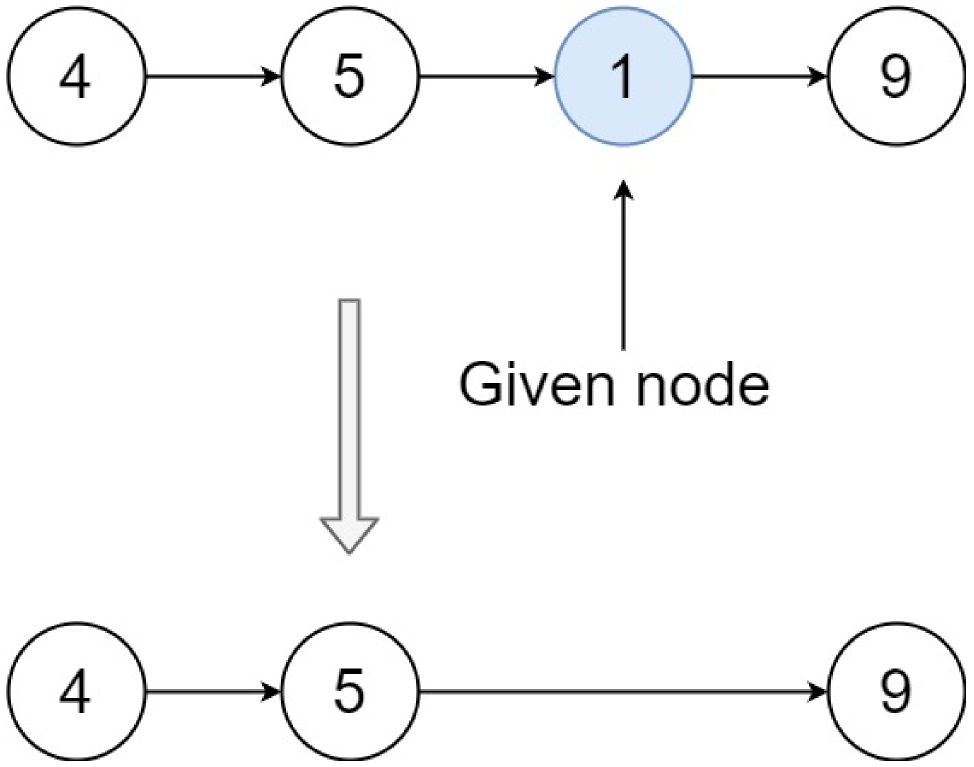
Output:

[4,1,9]

Explanation:

You are given the second node with value 5, the linked list should become 4 -> 1 -> 9 after calling your function.

Example 2:



Input:

head = [4,5,1,9], node = 1

Output:

[4,5,9]

Explanation:

You are given the third node with value 1, the linked list should become 4 -> 5 -> 9 after calling your function.

Constraints:

The number of the nodes in the given list is in the range

[2, 1000]

.

-1000 <= Node.val <= 1000

The value of each node in the list is

unique

.

The

node

to be deleted is

in the list

and is

not a tail

node.

Code Snippets

C++:

```
/**
* Definition for singly-linked list.
* struct ListNode {
* int val;
* ListNode *next;
* ListNode(int x) : val(x), next(NULL) {}
* };
*/
class Solution {
public:
void deleteNode(ListNode* node) {

}
};
```

Java:

```
/**
* Definition for singly-linked list.
* public class ListNode {
* int val;
* ListNode next;
* ListNode(int x) { val = x; }
* }
*/
class Solution {
public void deleteNode(ListNode node) {

}
}
```

Python3:

```
# Definition for singly-linked list.
# class ListNode:
# def __init__(self, x):
# self.val = x
# self.next = None

class Solution:
def deleteNode(self, node):
"""
:type node: ListNode
:rtype: void Do not return anything, modify node in-place instead.
"""
```

Python:

```
# Definition for singly-linked list.
# class ListNode(object):
# def __init__(self, x):
# self.val = x
# self.next = None

class Solution(object):
```

```
def deleteNode(self, node):
"""
:type node: ListNode
:rtype: void Do not return anything, modify node in-place instead.
"""
```

JavaScript:

```
/**
* Definition for singly-linked list.
* function ListNode(val) {
* this.val = val;
* this.next = null;
* }
*/
/**
* @param {ListNode} node
* @return {void} Do not return anything, modify node in-place instead.
*/
var deleteNode = function(node) {

};
```

TypeScript:

```
/**
* Definition for singly-linked list.
* class ListNode {
* val: number
* next: ListNode | null
* constructor(val?: number, next?: ListNode | null) {
* this.val = (val===undefined ? 0 : val)
* this.next = (next===undefined ? null : next)
* }
* }
*/

/**
Do not return anything, modify it in-place instead.
*/
function deleteNode(node: ListNode | null): void {

};
```

C#:

```
/**
* Definition for singly-linked list.
* public class ListNode {
* public int val;
* public ListNode next;
* public ListNode(int x) { val = x; }
* }
*/
public class Solution {
public void DeleteNode(ListNode node) {

}
}
```

C:

```
/**
* Definition for singly-linked list.
* struct ListNode {
* int val;
* struct ListNode *next;
* };
*/
void deleteNode(struct ListNode* node) {

}
```

Go:

```
/**
* Definition for singly-linked list.
* type ListNode struct {
* Val int
* Next *ListNode
* }
*/
func deleteNode(node *ListNode) {
}
```

Kotlin:

```
/**
* Example:
* var li = ListNode(5)
* var v = li.`val`
* Definition for singly-linked list.
* class ListNode(var `val`: Int) {
* var next: ListNode? = null
* }
*/

class Solution {
fun deleteNode(node: ListNode?) {

}
}
```

Swift:

```
/**
* Definition for singly-linked list.
* public class ListNode {
* public var val: Int
* public var next: ListNode?
* public init(_ val: Int) {
* self.val = val
* self.next = nil
* }
* }
*/

class Solution {
func deleteNode(_ node: ListNode?) {

}
}
```

Ruby:

```
# Definition for singly-linked list.
# class ListNode
# attr_accessor :val, :next
# def initialize(val)
# @val = val
# @next = nil
# end
# end

# @param {ListNode} node
# @return {Void} Do not return anything, modify node in-place instead.
def delete_node(node)

end
```

PHP:

```
/**
* Definition for a singly-linked list.
* class ListNode {
* public $val = 0;
* public $next = null;
* function __construct($val) { $this->val = $val; }
* }
*/

class Solution {
/**
* @param ListNode $node
* @return
*/
function deleteNode($node) {

}
}
```

Scala:

```
/**
* Definition for singly-linked list.
* class ListNode(var _x: Int = 0) {
* var next: ListNode = null
* var x: Int = _x
* }
*/

object Solution {
def deleteNode(node: ListNode): Unit = {

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Delete Node in a Linked List
* Difficulty: Medium
* Tags: linked_list
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* Definition for singly-linked list.
* struct ListNode {
* int val;
* ListNode *next;
* ListNode(int x) : val(x), next(NULL) {
// TODO: Implement optimized solution
return 0;
}
* };
*/
class Solution {
public:
void deleteNode(ListNode* node) {

}
};
```

Java Solution:

```
/**
* Problem: Delete Node in a Linked List
* Difficulty: Medium
* Tags: linked_list
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* Definition for singly-linked list.
* public class ListNode {
* int val;
* ListNode next;
* ListNode(int x) { val = x; }
* }
*/
class Solution {
public void deleteNode(ListNode node) {

}
}
```

Python3 Solution:

```
# Definition for singly-linked list.
# class ListNode:
# def __init__(self, x):
# self.val = x
# self.next = None

class Solution:
def deleteNode(self, node):
"""
:type node: ListNode
:rtype: void Do not return anything, modify node in-place instead.
"""
```

Python Solution:

```
# Definition for singly-linked list.
# class ListNode(object):
# def __init__(self, x):
# self.val = x
# self.next = None

class Solution(object):
def deleteNode(self, node):
"""
:type node: ListNode
:rtype: void Do not return anything, modify node in-place instead.
"""
```

JavaScript Solution:

```
/**
* Problem: Delete Node in a Linked List
* Difficulty: Medium
* Tags: linked_list
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* Definition for singly-linked list.
* function ListNode(val) {
* this.val = val;
* this.next = null;
* }
*/
/**
* @param {ListNode} node
* @return {void} Do not return anything, modify node in-place instead.
*/
var deleteNode = function(node) {

};
```

TypeScript Solution:

```
/**
* Problem: Delete Node in a Linked List
* Difficulty: Medium
* Tags: linked_list
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* Definition for singly-linked list.
* class ListNode {
* val: number
* next: ListNode | null
* constructor(val?: number, next?: ListNode | null) {
* this.val = (val===undefined ? 0 : val)
* this.next = (next===undefined ? null : next)
* }
* }
*/

/**
Do not return anything, modify it in-place instead.
*/
function deleteNode(node: ListNode | null): void {

};
```

C# Solution:

```
/*
* Problem: Delete Node in a Linked List
* Difficulty: Medium
* Tags: linked_list
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
```

```
* Definition for singly-linked list.
* public class ListNode {
* public int val;
* public ListNode next;
* public ListNode(int x) { val = x; }
* }
*/
public class Solution {
public void DeleteNode(ListNode node) {

}
}
```

C Solution:

```
/*
* Problem: Delete Node in a Linked List
* Difficulty: Medium
* Tags: linked_list
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* Definition for singly-linked list.
* struct ListNode {
* int val;
* struct ListNode *next;
* };
*/
void deleteNode(struct ListNode* node) {

}
```

Go Solution:

```
// Problem: Delete Node in a Linked List
// Difficulty: Medium
// Tags: linked_list
```

```
//
// Approach: Optimized algorithm based on problem constraints
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

/**
* Definition for singly-linked list.
* type ListNode struct {
* Val int
* Next *ListNode
* }
*/
func deleteNode(node *ListNode) {

}
```

Kotlin Solution:

```
/**
* Example:
* var li = ListNode(5)
* var v = li.`val`
* Definition for singly-linked list.
* class ListNode(var `val`: Int) {
* var next: ListNode? = null
* }
*/

class Solution {
fun deleteNode(node: ListNode?) {

}
}
```

Swift Solution:

```
/**
* Definition for singly-linked list.
* public class ListNode {
* public var val: Int
* public var next: ListNode?
* public init(_ val: Int) {
* self.val = val
* self.next = nil
* }
* }
*/

class Solution {
func deleteNode(_ node: ListNode?) {

}
}
```

Ruby Solution:

```
# Definition for singly-linked list.
# class ListNode
# attr_accessor :val, :next
# def initialize(val)
# @val = val
# @next = nil
# end
# end

# @param {ListNode} node
# @return {Void} Do not return anything, modify node in-place instead.
def delete_node(node)

end
```

PHP Solution:

```
/**
* Definition for a singly-linked list.
* class ListNode {
* public $val = 0;
* public $next = null;
* function __construct($val) { $this->val = $val; }
* }
*/
```

```
class Solution {
/**
* @param ListNode $node
* @return
*/
function deleteNode($node) {

}
}
```

Scala Solution:

```
/**
* Definition for singly-linked list.
* class ListNode(var _x: Int = 0) {
* var next: ListNode = null
* var x: Int = _x
* }
*/

object Solution {
def deleteNode(node: ListNode): Unit = {

}
}
```