

## Problem 1721: Swapping Nodes in a Linked List

## Problem Information

## Difficulty: Medium

**Acceptance Rate:** 0.00%

**Paid Only: No**

## Problem Description

You are given the

head

of a linked list, and an integer

k

## Return

the head of the linked list after

## swapping

the values of the

4

th

node from the beginning and the

4

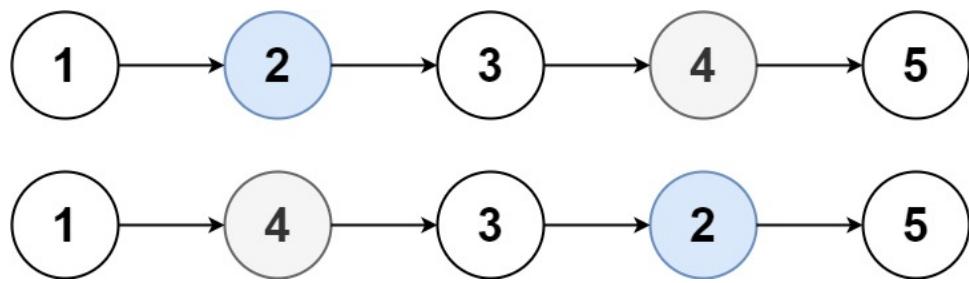
th

node from the end (the list is

1-indexed

).

Example 1:



Input:

head = [1,2,3,4,5], k = 2

Output:

[1,4,3,2,5]

Example 2:

Input:

head = [7,9,6,6,7,8,3,0,9,5], k = 5

Output:

[7,9,6,6,8,7,3,0,9,5]

Constraints:

The number of nodes in the list is

n

1 <= k <= n <= 10

5

0 <= Node.val <= 100

## Code Snippets

### C++:

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* swapNodes(ListNode* head, int k) {
        }

    };
}
```

### Java:

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 * }
```

```

* ListNode(int val, ListNode next) { this.val = val; this.next = next; }
* }
*/
class Solution {
public ListNode swapNodes(ListNode head, int k) {

}
}

```

### Python3:

```

# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def swapNodes(self, head: Optional[ListNode], k: int) -> Optional[ListNode]:

```

### Python:

```

# Definition for singly-linked list.
# class ListNode(object):
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution(object):
    def swapNodes(self, head, k):
        """
        :type head: Optional[ListNode]
        :type k: int
        :rtype: Optional[ListNode]
        """

```

### JavaScript:

```

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

```

```

/*
/**
 * @param {ListNode} head
 * @param {number} k
 * @return {ListNode}
 */
var swapNodes = function(head, k) {

};

```

### 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)
 *   }
 * }
 */

function swapNodes(head: ListNode | null, k: number): ListNode | null {
}

;

```

### C#:

```

/**
 * Definition for singly-linked list.
 * public class ListNode {
 *   public int val;
 *   public ListNode next;
 *   public ListNode(int val=0, ListNode next=null) {
 *     this.val = val;
 *     this.next = next;
 *   }
 * }
 */

public class Solution {

```

```
public ListNode SwapNodes(ListNode head, int k) {  
    }  
    }  
}
```

## C:

```
/**  
 * Definition for singly-linked list.  
 * struct ListNode {  
 *     int val;  
 *     struct ListNode *next;  
 * };  
 */  
struct ListNode* swapNodes(struct ListNode* head, int k) {  
  
}
```

## Go:

```
/**  
 * Definition for singly-linked list.  
 * type ListNode struct {  
 *     Val int  
 *     Next *ListNode  
 * }  
 */  
func swapNodes(head *ListNode, k int) *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 swapNodes(head: ListNode?, k: Int): ListNode? {  
        }  
        }  
}
```

## Swift:

```
/**  
 * Definition for singly-linked list.  
 *  
 * public class ListNode {  
 *     public var val: Int  
 *     public var next: ListNode?  
 *     public init() { self.val = 0; self.next = nil; }  
 *     public init(_ val: Int) { self.val = val; self.next = nil; }  
 *     public init(_ val: Int, _ next: ListNode?) { self.val = val; self.next =  
 *         next; }  
 * }  
 */  
class Solution {  
    func swapNodes(_ head: ListNode?, _ k: Int) -> ListNode? {  
        }  
        }  
}
```

## Rust:

```
// Definition for singly-linked list.  
// #[derive(PartialEq, Eq, Clone, Debug)]  
// pub struct ListNode {  
//     pub val: i32,  
//     pub next: Option<Box<ListNode>>  
// }  
//  
// impl ListNode {  
//     #[inline]  
//     fn new(val: i32) -> Self {  
//         ListNode {  
//             next: None,  
//             val  
//         }  
//     }  
// }
```

```

    // }

impl Solution {
    pub fn swap_nodes(head: Option<Box<ListNode>>, k: i32) ->
        Option<Box<ListNode>> {
        }

    }
}

```

### Ruby:

```

# Definition for singly-linked list.
# class ListNode
# attr_accessor :val, :next
# def initialize(val = 0, _next = nil)
#   @val = val
#   @_next = _next
# end
# end
# @param {ListNode} head
# @param {Integer} k
# @return {ListNode}
def swap_nodes(head, k)

end

```

### PHP:

```

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

/**
 * @param ListNode $head

```

```

* @param Integer $k
* @return ListNode
*/
function swapNodes($head, $k) {
}
}

```

### Dart:

```

/**
 * Definition for singly-linked list.
 * class ListNode {
 * int val;
 * ListNode? next;
 * ListNode([this.val = 0, this.next]);
 * }
 */
class Solution {
ListNode? swapNodes(ListNode? head, int k) {
}

}

```

### Scala:

```

/**
 * Definition for singly-linked list.
 * class ListNode(_x: Int = 0, _next: ListNode = null) {
 * var next: ListNode = _next
 * var x: Int = _x
 * }
 */
object Solution {
def swapNodes(head: ListNode, k: Int): ListNode = {

}
}

```

### Elixir:

```

# Definition for singly-linked list.

#
# defmodule ListNode do
# @type t :: %__MODULE__{
#   val: integer,
#   next: ListNode.t() | nil
# }
# defstruct val: 0, next: nil
# end

defmodule Solution do
@spec swap_nodes(ListNode.t() | nil, integer()) :: ListNode.t() | nil
def swap_nodes(head, k) do

end
end

```

### Erlang:

```

%% Definition for singly-linked list.

%%
%% -record(list_node, {val = 0 :: integer(),
%% next = null :: 'null' | #list_node{}}).

-spec swap_nodes(Head :: #list_node{} | null, K :: integer()) -> #list_node{} |
null.
swap_nodes(Head, K) ->
.
.
```

### Racket:

```

; Definition for singly-linked list:
#| 

; val : integer?
; next : (or/c list-node? #f)
(struct list-node
  (val next) #:mutable #:transparent)

; constructor
(define (make-list-node [val 0])
  (list-node val #f))

```

```

| #

(define/contract (swap-nodes head k)
  (-> (or/c list-node? #f) exact-integer? (or/c list-node? #f)))
)

```

## Solutions

### C++ Solution:

```

/*
 * Problem: Swapping Nodes in a Linked List
 * Difficulty: Medium
 * Tags: array, linked_list
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* swapNodes(ListNode* head, int k) {

    }
};

```

### Java Solution:

```

/**
 * Problem: Swapping Nodes in a Linked List
 * Difficulty: Medium
 * Tags: array, linked_list
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     // TODO: Implement optimized solution
 *     return 0;
 * }
 * ListNode(int val) { this.val = val; }
 * ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 */
class Solution {
    public ListNode swapNodes(ListNode head, int k) {
}
}

```

### Python3 Solution:

```

"""
Problem: Swapping Nodes in a Linked List
Difficulty: Medium
Tags: array, linked_list

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

# Definition for singly-linked list.

```

```

# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
#
# class Solution:
#     def swapNodes(self, head: Optional[ListNode], k: int) -> Optional[ListNode]:
#         # TODO: Implement optimized solution
#         pass

```

### Python Solution:

```

# Definition for singly-linked list.
# class ListNode(object):
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
#
# class Solution(object):
#     def swapNodes(self, head, k):
#         """
# :type head: Optional[ListNode]
# :type k: int
# :rtype: Optional[ListNode]
#         """

```

### JavaScript Solution:

```

/**
 * Problem: Swapping Nodes in a Linked List
 * Difficulty: Medium
 * Tags: array, linked_list
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

/**
 * Definition for singly-linked list.
 * function ListNode(val, next) {
*     this.val = (val===undefined ? 0 : val)
*     this.next = (next===undefined ? null : next)

```

```

* }
*/
/** 
 * @param {ListNode} head
 * @param {number} k
 * @return {ListNode}
 */
var swapNodes = function(head, k) {
};

}

```

### TypeScript Solution:

```

/** 
 * Problem: Swapping Nodes in a Linked List
 * Difficulty: Medium
 * Tags: array, linked_list
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * 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)
 *   }
 * }
 */

function swapNodes(head: ListNode | null, k: number): ListNode | null {

}

```

### C# Solution:

```

/*
 * Problem: Swapping Nodes in a Linked List
 * Difficulty: Medium
 * Tags: array, linked_list
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * 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 val=0, ListNode next=null) {
 *         this.val = val;
 *         this.next = next;
 *     }
 * }
 */
public class Solution {
    public ListNode SwapNodes(ListNode head, int k) {
        }

    }
}

```

## C Solution:

```

/*
 * Problem: Swapping Nodes in a Linked List
 * Difficulty: Medium
 * Tags: array, linked_list
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

/**
 * Definition for singly-linked list.
 * struct ListNode {

```

```

* int val;
* struct ListNode *next;
* };
*/
struct ListNode* swapNodes(struct ListNode* head, int k) {
}

```

### Go Solution:

```

// Problem: Swapping Nodes in a Linked List
// Difficulty: Medium
// Tags: array, linked_list
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

/**
* Definition for singly-linked list.
* type ListNode struct {
* Val int
* Next *ListNode
* }
*/
func swapNodes(head *ListNode, k int) *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 swapNodes(head: ListNode?, k: Int): ListNode? {  
    }  
    }  
}
```

### Swift Solution:

```
/**  
 * Definition for singly-linked list.  
 *  
 * public class ListNode {  
 *     public var val: Int  
 *     public var next: ListNode?  
 *     public init() { self.val = 0; self.next = nil; }  
 *     public init(_ val: Int) { self.val = val; self.next = nil; }  
 *     public init(_ val: Int, _ next: ListNode?) { self.val = val; self.next =  
 *         next; }  
 * }  
 */  
class Solution {  
    func swapNodes(_ head: ListNode?, _ k: Int) -> ListNode? {  
        }  
    }
```

### Rust Solution:

```
// Problem: Swapping Nodes in a Linked List  
// Difficulty: Medium  
// Tags: array, linked_list  
//  
// Approach: Use two pointers or sliding window technique  
// Time Complexity: O(n) or O(n log n)  
// Space Complexity: O(1) to O(n) depending on approach  
  
// Definition for singly-linked list.  
// #[derive(PartialEq, Eq, Clone, Debug)]  
// pub struct ListNode {  
//     pub val: i32,  
//     pub next: Option<Box<ListNode>>  
// }  
//
```

```

// impl ListNode {
// #[inline]
// fn new(val: i32) -> Self {
// ListNode {
// next: None,
// val
// }
// }
// }

impl Solution {
pub fn swap_nodes(head: Option<Box<ListNode>>, k: i32) ->
Option<Box<ListNode>> {

}
}

```

### Ruby Solution:

```

# Definition for singly-linked list.
# class ListNode
# attr_accessor :val, :next
# def initialize(val = 0, _next = nil)
#   @val = val
#   @next = _next
# end
# end
# @param {ListNode} head
# @param {Integer} k
# @return {ListNode}
def swap_nodes(head, k)

end

```

### PHP Solution:

```

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

```

```

* $this->val = $val;
* $this->next = $next;
* }
* }
*/
class Solution {

/**
* @param ListNode $head
* @param Integer $k
* @return ListNode
*/
function swapNodes($head, $k) {
}

}
}

```

### Dart Solution:

```

/**
* Definition for singly-linked list.
* class ListNode {
* int val;
* ListNode? next;
* ListNode([this.val = 0, this.next]);
* }
*/
class Solution {
ListNode? swapNodes(ListNode? head, int k) {
}

}
}

```

### Scala Solution:

```

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

```

```

*/
object Solution {
def swapNodes(head: ListNode, k: Int): ListNode = {

}
}

```

### Elixir Solution:

```

# Definition for singly-linked list.
#
# defmodule ListNode do
# @type t :: %__MODULE__{
#   val: integer,
#   next: ListNode.t() | nil
# }
# defstruct val: 0, next: nil
# end

defmodule Solution do
@spec swap_nodes(ListNode.t() | nil, integer) :: ListNode.t() | nil
def swap_nodes(head, k) do

end
end

```

### Erlang Solution:

```

%% Definition for singly-linked list.
%%
%% -record(list_node, {val = 0 :: integer(),
%% next = null :: 'null' | #list_node{}}).

-spec swap_nodes(Head :: #list_node{} | null, K :: integer()) -> #list_node{} | null.
swap_nodes(Head, K) ->
.
```

### Racket Solution:

```
; Definition for singly-linked list:  
#|  
  
; val : integer?  
; next : (or/c list-node? #f)  
(struct list-node  
(val next) #:mutable #:transparent)  
  
; constructor  
(define (make-list-node [val 0])  
(list-node val #f))  
  
|#  
  
(define/contract (swap-nodes head k)  
(-> (or/c list-node? #f) exact-integer? (or/c list-node? #f))  
)
```