package com.iimtiaz.day\_12;  
  
  
public class ReverseLinkedList {  
 public static void main(String[] args) {  
 // Create a linked list  
 ListNode head = new ListNode(1);  
 head.next = new ListNode(2);  
 head.next.next = new ListNode(3);  
 head.next.next.next = new ListNode(4);  
  
 // Print the original linked list  
 *printList*(head);  
  
 // Call the reverseList method  
 Solution solution = new Solution();  
 ListNode newHead = solution.reverseList(head);  
  
 // Print the reversed linked list  
 *printList*(newHead);  
 }  
  
 // Helper method to print a linked list  
 public static void printList(ListNode head) {  
 ListNode current = head;  
 while (current != null) {  
 System.*out*.print(current.val + " ");  
 current = current.next;  
 }  
 System.*out*.println();  
 }  
}  
  
*/\*\*  
 \* Time Complexity: O(n)  
 \* <p>  
 \* O(n): The while loop iterates through each node in the linked list once, ensuring linear time complexity.  
 \* O(1): Operations within the loop (pointer manipulations) are constant time.  
 \* Space Complexity: O(1)  
 \* <p>  
 \* Uses a few fixed-size variables (prev, current, next), regardless of the input linked list size.  
 \* No additional data structures that grow with input size are created.  
 \*/*class Solution {  
 public ListNode reverseList(ListNode head) {  
 ListNode prev = null;  
 ListNode current = head;  
 while (current != null) {  
 ListNode next = current.next;  
 current.next = prev;  
 prev = current;  
 current = next;  
 }  
 return prev;  
 }  
}  
  
  
class ListNode {  
 int val;  
 ListNode next;  
  
 ListNode() {  
 }  
  
 ListNode(int val) {  
 this.val = val;  
 }  
  
 ListNode(int val, ListNode next) {  
 this.val = val;  
 this.next = next;  
 }  
}  
  
// https://leetcode.com/problems/reverse-linked-list/description/