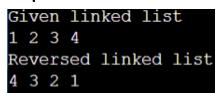


#### **Output:**



## Approach:

- The idea is to use three pointers current, previous, and next to keep track of nodes to update reverse links.
- Initialize three pointers previous as NULL, current as head, and next as NULL.
- Iterate through the linked list. In a loop, do the following:
- Before changing the next of current, store the next node
  - next = current -> next
  - Now update the next pointer of curr to the prev
  - current -> next = previous
  - Update prev as curr and curr as next
  - previous = current
  - current = next

# **Recursive approach:**

Code: LP\_Code6.java

# Output:

Given linked list 4 3 2 1 Reversed linked list 1 2 3 4

#### Approach:

- The main concept of using recursion to reverse a linked list is to reach the last node of the linked list using recursion then start reversing the linked list.
- Divide the list in two parts first node and rest of the linked list.
- · Recursively call reverse for the rest of the linked list.
- Link the rest linked list to first and fix the head pointer to NULL.

# Middle of a linked list:

Question: Given the head of a singly linked list, return the middle node of the linked list. If there are two middle nodes, return the second middle node.

# Example 1:



**Input:** head = [1,2,3,4,5]

Output: 3

**Explanation:** The middle node of the list is node 3.

### Example 2:

