

Leetcode Problem - 142

Code in java:

```
public class Solution {
    public ListNode detectCycle(ListNode head) {
        if (head == null || head.next == null) return null;

        ListNode slow = head;
        ListNode fast = head;

        while (fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;

            if (slow == fast) {
                ListNode entry = head;
                while (entry != slow) {
                    entry = entry.next;
                    slow = slow.next;
                }
                return entry;
            }
        }
        return null;
    }
}
```

The screenshot shows a LeetCode submission interface. The top bar includes the Safari browser menu and system status (19°C, 92% battery, 11:14:32 AM). The LeetCode header shows the problem name 'Detect Cycle in a Linked List' and the user 'Janish_805' who submitted at Jan 20, 2026 11:14. The submission status is 'Accepted' with 28/28 test cases passed. The runtime is 0 ms (100.00% beats) and memory is 44.41 MB (19.20% beats). The code is in Java and implements Floyd's Cycle-Finding algorithm. The test result shows Case 1, Case 2, and Case 3 all passed with input [1, 2, 3, 4, 5].

```
9  /**
10  */
11  class Solution {
12      public ListNode reverseList(ListNode head) {
13          ListNode prev = null;
14          ListNode curr = head;
15
16          while (curr != null) {
17              ListNode next = curr.next;
18              curr.next = prev;
19              prev = curr;
20              curr = next;
21          }
22          return prev;
23      }
24  }
25
```

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

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