

PROBLEM:-142

```
class Solution {  
  
public:  
  
    ListNode *detectCycle(ListNode *head) {  
  
        if (!head || !head->next) return NULL;  
  
        ListNode *slow = head, *fast = head;  
  
        while (fast && fast->next) {  
  
            slow = slow->next;  
  
            fast = fast->next->next;  
  
            if (slow == fast) break;  
  
        }  
  
        if (!fast || !fast->next) return NULL;  
  
        slow = head;  
  
        while (slow != fast) {  
  
            slow = slow->next;  
  
            fast = fast->next;  
  
        }  
  
        return slow;  
    }  
};
```

C++ Auto

```
1 class Solution {
2 public:
3     ListNode *detectCycle(ListNode *head) {
4         if (!head || !head->next) return NULL;
5         ListNode *slow = head, *fast = head;
6         while (fast && fast->next) {
7             slow = slow->next;
8             fast = fast->next->next;
9             if (slow == fast) break;
10        }
11        if (!fast || !fast->next) return NULL;
12        slow = head;
```

Saved

Testcase | Test Result

Accepted Runtime: 3 ms

Case 1

Case 2

Case 3

Input

head =

[3,2,0,-4]

pos =

1

PROBLEM:- 206

```
class Solution {  
  
public:  
  
    ListNode* reverseList(ListNode* head) {  
  
        ListNode* prev = nullptr;  
  
        ListNode* curr = head;  
  
        while (curr) {  
  
            ListNode* nextNode = curr->next;  
  
            curr->next = prev;  
  
            prev = curr;  
  
            curr = nextNode;  
  
        }  
  
        return prev;  
  
    };
```

</> Code

C++ Auto

```
5     ListNode* curr = head;
6
7     while (curr) {
8         ListNode* nextNode = curr->next;
9         curr->next = prev;
10        prev = curr;
11        curr = nextNode;
12    }
13    return prev;
14}
15}
16
```

Saved

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1

Case 2

Case 3

Input

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

Output

```
[5,4,3,2,1]
```

Expected