

### Day – 5 : Linked List -I

**Problem 1:** Given the *head* of a singly linked list, write a program to reverse the linked list, and return *the head pointer to the reversed list*.

```
class ListNode:
```

```
    def __init__(self, val=0, next=None):
```

```
        self.val = val
```

```
        self.next = next
```

```
def reverseLinkedList(head):
```

```
    prev = None
```

```
    current = head
```

```
    while current is not None:
```

```
        next_node = current.next
```

```
        current.next = prev
```

```
        prev = current
```

```
        current = next_node
```

```
    return prev
```

```
# Test the program
```

```
def createLinkedList(arr):
```

```
    head = ListNode(arr[0])
```

```
    current = head
```

```
    for i in range(1, len(arr)):
```

```
        current.next = ListNode(arr[i])
```

```
        current = current.next
```

```
    return head
```

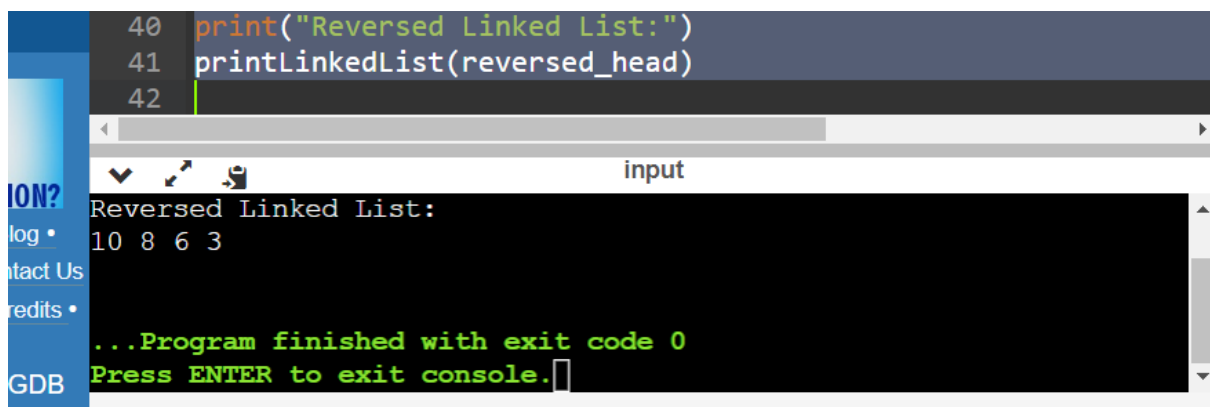
```
def printLinkedList(head):
```

```
    current = head
```

```
while current is not None:
    print(current.val, end=" ")
    current = current.next
print()
```

```
arr = [3, 6, 8, 10]
head = createLinkedList(arr)
print("Original Linked List:")
printLinkedList(head)
```

```
reversed_head = reverseLinkedList(head)
print("Reversed Linked List:")
printLinkedList(reversed_head)
```

A screenshot of a code editor and terminal. The code editor shows three lines of Python code: line 40: print("Reversed Linked List:"); line 41: printLinkedList(reversed\_head); line 42: (empty line). The terminal window below shows the output: "Reversed Linked List:" followed by "10 8 6 3" on the next line. Below the output, it says "...Program finished with exit code 0" and "Press ENTER to exit console." with a cursor. The terminal window has a title bar that says "input".

```
40 print("Reversed Linked List:")
41 printLinkedList(reversed_head)
42
Reversed Linked List:
10 8 6 3
...Program finished with exit code 0
Press ENTER to exit console.
```

**Problem 2:** 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.

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
```

```
def find_middle_node(head):
```

```
if not head:  
    return None
```

```
slow = head
```

```
fast = head
```

```
while fast and fast.next:
```

```
    slow = slow.next
```

```
    fast = fast.next.next
```

```
return slow
```

```
def list_to_linked_list(lst):
```

```
    dummy_head = ListNode()
```

```
    current = dummy_head
```

```
    for val in lst:
```

```
        current.next = ListNode(val)
```

```
        current = current.next
```

```
    return dummy_head.next
```

```
def linked_list_to_list(head):
```

```
    lst = []
```

```
    current = head
```

```
    while current:
```

```
        lst.append(current.val)
```

```
        current = current.next
```

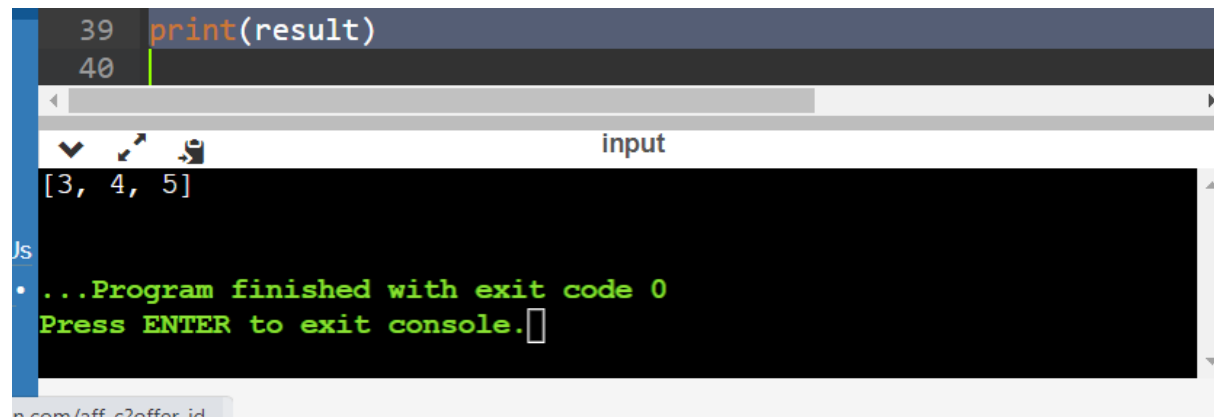
```
    return lst
```

```
head = list_to_linked_list([1, 2, 3, 4, 5])
```

```
middle_node = find_middle_node(head)
```

```
result = linked_list_to_list(middle_node)

print(result)
```

A screenshot of a code editor and terminal. The code editor shows two lines of Python code: line 39 with `print(result)` and line 40 which is empty. Below the code editor is a terminal window. The terminal has a title bar that says "input". Inside the terminal, the input `[3, 4, 5]` is shown. Below the input, the terminal displays the message `...Program finished with exit code 0` and `Press ENTER to exit console.` with a cursor. At the bottom of the terminal window, a URL `n.com/off_c?offer_id` is partially visible.

```
39 print(result)
40
input
[3, 4, 5]
...Program finished with exit code 0
Press ENTER to exit console.
n.com/off_c?offer_id
```

---

**Problem 3:** Given two singly linked lists that are sorted in increasing order of node values, merge two **sorted** linked lists and return them as a sorted list. The list should be made by splicing together the nodes of the first two lists.

```
class ListNode:
```

```
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
```

```
def mergeTwoLists(l1, l2):
```

```
    dummy = ListNode(-1)
    current = dummy
```

```
    while l1 and l2:
```

```
        if l1.val <= l2.val:
            current.next = l1
            l1 = l1.next
        else:
            current.next = l2
            l2 = l2.next
```

```
current = current.next
```

```
current.next = l1 if l1 else l2
```

```
return dummy.next
```

```
def printLinkedList(head):
```

```
    result = []
```

```
    while head:
```

```
        result.append(head.val)
```

```
        head = head.next
```

```
    return result
```

```
l1 = ListNode(3)
```

```
l1.next = ListNode(7)
```

```
l1.next.next = ListNode(10)
```

```
l2 = ListNode(1)
```

```
l2.next = ListNode(2)
```

```
l2.next.next = ListNode(5)
```

```
l2.next.next.next = ListNode(8)
```

```
l2.next.next.next.next = ListNode(10)
```

```
merged_list = mergeTwoLists(l1, l2)
```

```
print(printLinkedList(merged_list))
```

```
18
19     current.next = 11 if 11 else 12
20
21     return dummy.next
22
```

input

```
[1, 2, 3, 5, 7, 8, 10, 10]
```

```
...Program finished with exit code 0
Press ENTER to exit console.
```

3 GDB

**Problem 4:** Given a [linked list](#), and a number N. Find the Nth node from the end of this linked list and delete it. Return the head of the new modified linked list.

class ListNode:

```
def __init__(self, val=0, next=None):
    self.val = val
    self.next = next
```

def remove\_nth\_from\_end(head, n):

```
    first = head
    second = head
```

```
    for i in range(n):
```

```
        if first.next:
            first = first.next
        else:
```

```
            return head
```

```
    while first.next:
```

```
        first = first.next
```

```
    second = second.next
```

```
if not second.next:
```

```
    return head.next
```

```
else:
```

```
    second.next = second.next.next
```

```
return head
```

```
def list_to_linked_list(lst):
```

```
    if not lst:
```

```
        return None
```

```
    head = ListNode(lst[0])
```

```
    current = head
```

```
    for val in lst[1:]:
```

```
        current.next = ListNode(val)
```

```
        current = current.next
```

```
    return head
```

```
def linked_list_to_list(head):
```

```
    lst = []
```

```
    current = head
```

```
    while current:
```

```
        lst.append(current.val)
```

```
        current = current.next
```

```
    return lst
```

```

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

n = 2

head = list_to_linked_list(input_list)

new_head = remove_nth_from_end(head, n)

result_list = linked_list_to_list(new_head)

print(result_list)

```

```

59 print(result_list)
60
input
[1, 2, 3, 5]
...Program finished with exit code 0
Press ENTER to exit console.

```

**Problem 5:** Given the **heads** of two non-empty linked lists representing two non-negative integers. The digits are stored in **reverse order**, and each of their nodes contains a single digit. Add the two numbers and return the **sum** as a linked list.

```
class ListNode:
```

```
    def __init__(self, val=0, next=None):
```

```
        self.val = val
```

```
        self.next = next
```

```
def addTwoNumbers(l1, l2):
```

```
    dummy = ListNode()
```

```
    curr = dummy
```

```
    carry = 0
```

```
    p1, p2 = l1, l2
```



```
while p1 or p2:
```

```
    x = p1.val if p1 else 0
```

```
    y = p2.val if p2 else 0
```

```
    _sum = x + y + carry
```

```
    carry = _sum // 10
```

```
    curr.next = ListNode(_sum % 10)
```

```
    curr = curr.next
```

```
    p1 = p1.next if p1 else None
```

```
    p2 = p2.next if p2 else None
```

```
if carry:
```

```
    curr.next = ListNode(carry)
```

```
return dummy.next
```

```
l1 = ListNode(2)
```

```
l1.next = ListNode(4)
```

```
l1.next.next = ListNode(3)
```

```
l2 = ListNode(5)
```

```
l2.next = ListNode(6)
```

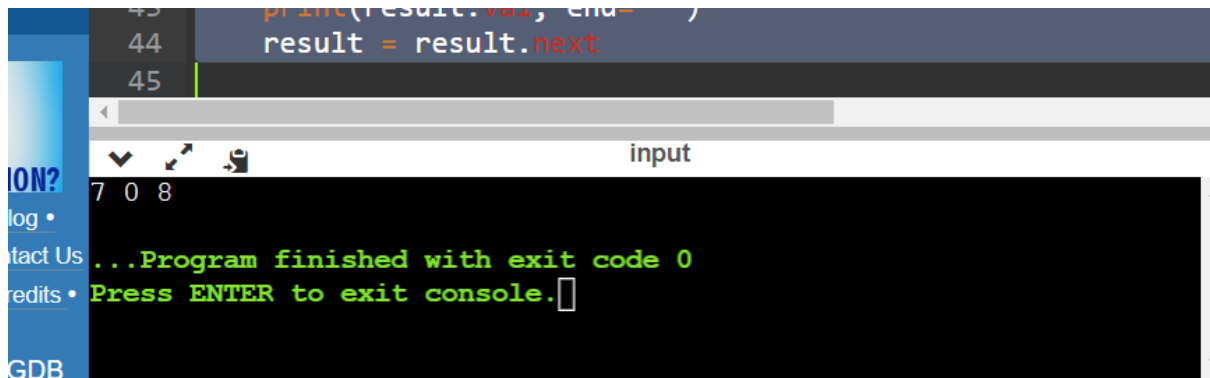
```
l2.next.next = ListNode(4)
```

```
result = addTwoNumbers(l1, l2)
```

```
while result:
```

```
print(result.val, end=" ")
```

```
result = result.next
```



```
43 print(result.val, end=" ")
44 result = result.next
45
```

input

7 0 8

...Program finished with exit code 0  
Press ENTER to exit console.

**Problem 6:** Write a function to **delete a node** in a singly-linked list. You will **not** be given access to the head of the list instead, you will be given access to **the node to be deleted** directly. It is **guaranteed** that the node to be deleted is **not a tail node** in the list.

```
class ListNode:
```

```
    def __init__(self, val=0, next=None):
```

```
        self.val = val
```

```
        self.next = next
```

```
def deleteNode(node):
```

```
    node.val = node.next.val
```

```
    node.next = node.next.next
```

```
node1 = ListNode(1)
```

```
node2 = ListNode(4)
```

```
node3 = ListNode(2)
```

```
node4 = ListNode(3)
```

```
node1.next = node2
```

```
node2.next = node3
```

```
node3.next = node4
```

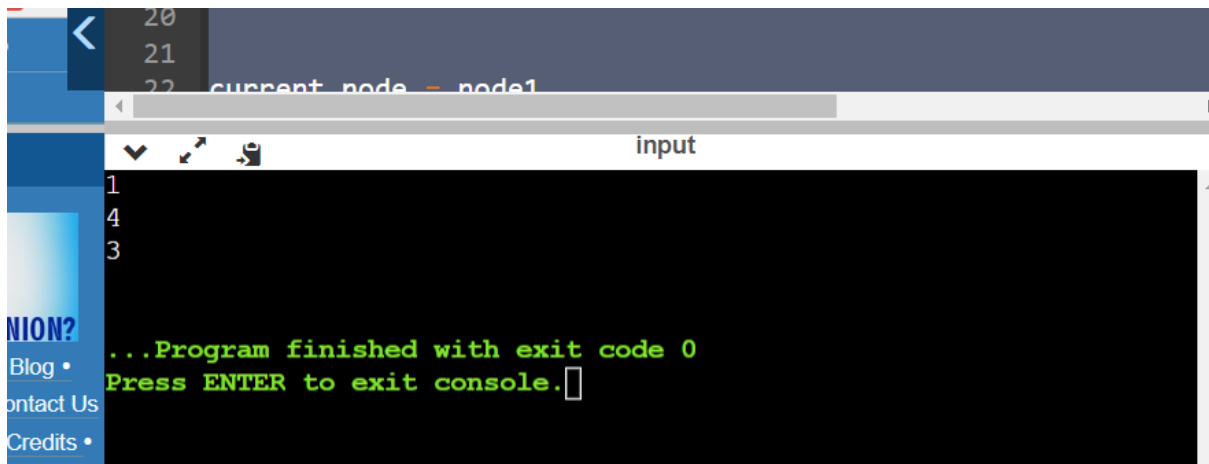
```
deleteNode(node3)
```

```
current_node = node1
```

```
while current_node:
```

```
    print(current_node.val)
```

```
    current_node = current_node.next
```



The screenshot shows a code editor with a dark theme. The code editor has a line number column on the left with numbers 20, 21, and 22. The code is as follows:

```
20  
21  
22 current_node = node1
```

Below the code editor is a terminal window titled "input". The terminal has a black background with green text. It shows the output of the program:

```
1  
4  
3  
...Program finished with exit code 0  
Press ENTER to exit console.
```

On the left side of the terminal window, there is a sidebar with a blue header "NION?" and a list of links: "Blog •", "Contact Us", and "Credits •".