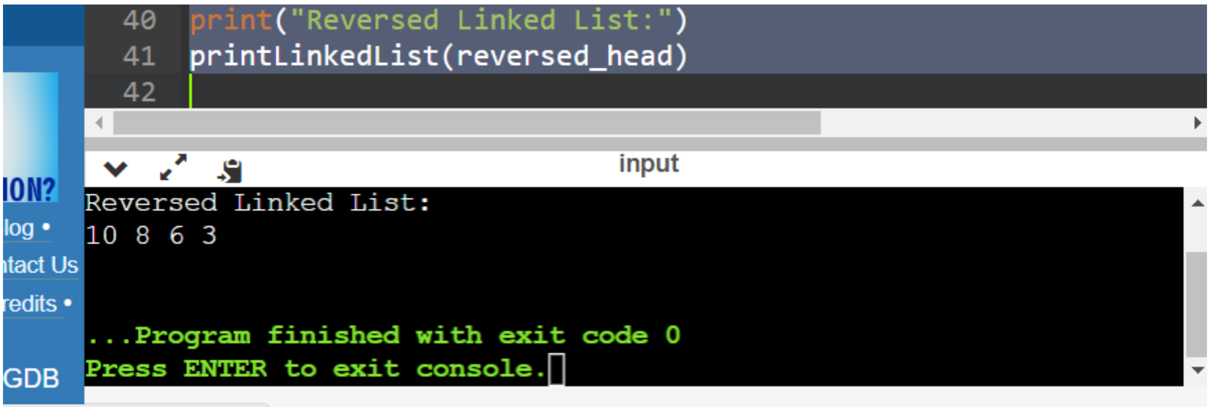
Name=Saransh Mittal
RollNo=2000290130145

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)
```



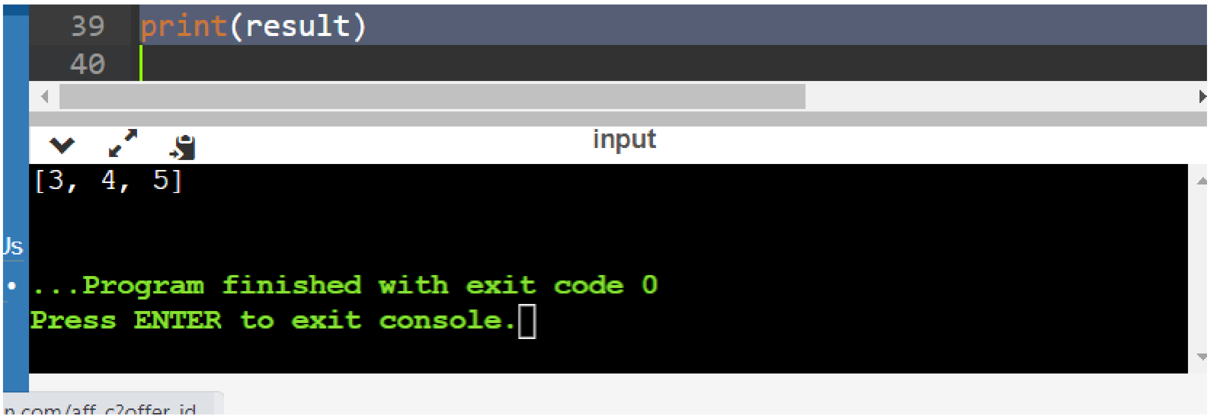
```
print("Reversed Linked List:")
printLinkedList(reversed_head)
```

```
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)
```

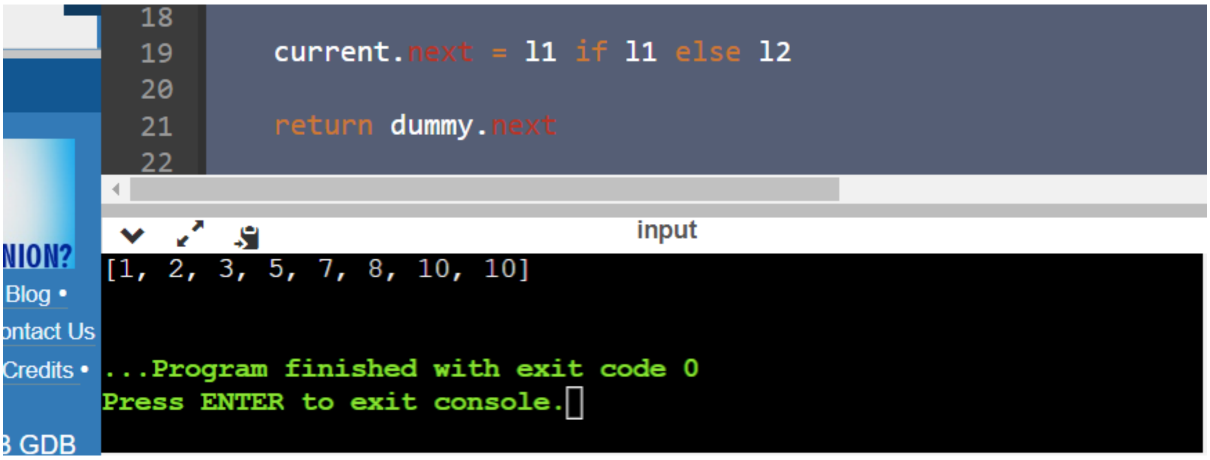


```
print(result)
```

```
[3, 4, 5]


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

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))
```



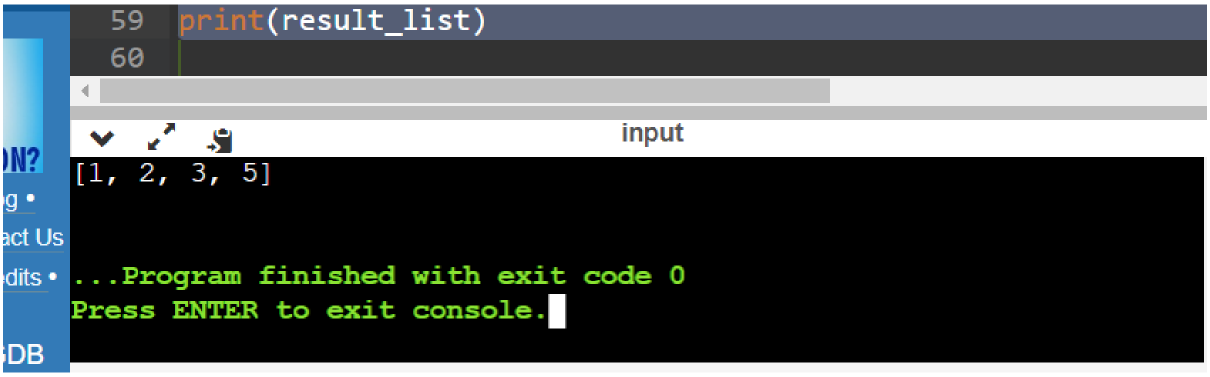
```python
        current.next = l1 if l1 else l2

        return dummy.next
```

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


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

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)
```



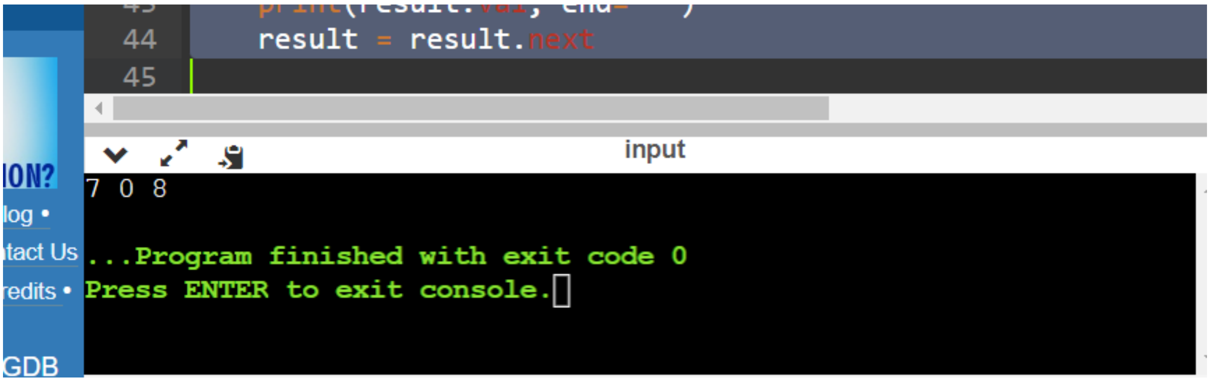
```
print(result_list)
```

```
[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
```



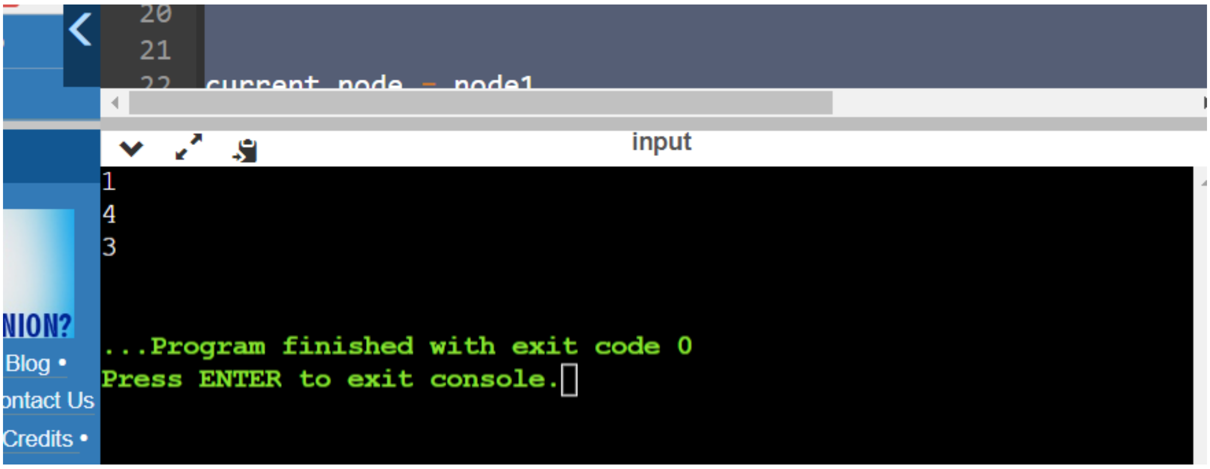
```
    result = result.next
```

```
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
```



```
current_node = node1
```

```
1
4
3


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