b) Linked List

The link shows a program to find the nth element of a linked list.

https://www.geeksforgeeks.org/nth-node-from-the-end-of-a-linked-list/

Find a way to find the kth to the last element of linked list (assume length of linked list is not known)

Bonus 1:

Can you minimize the number of times you run through the loop.

Solution:

```
class Node:
    def __init__(self, new_data):
        self.data = new_data
        self.next = None


class LinkedList:
    def __init__(self):
        self.head = None

    # CreateNode and make linked list in reverse order
    def push(self, new_data):
        # Create a new node with the given data
        new_node = Node(new_data)

        # If the linked list is empty, set the new node as the head
        if self.head is None:
            self.head = new_node
        else:
            # Find the last node
            last_node = self.head
            while last_node.next:
                last_node = last_node.next

            # Append the new node at the end
            last_node.next = new_node

    # Function to get the kth to the last element of a linked list
    def getKthToLast(self, k):
        if k <= 0:
            print("Invalid k. Please enter a positive integer.")
            return

        # Initialize two pointers, slow_ptr and fast_ptr, to the head of the
linked list
        slow_ptr = self.head
```

```
        fast_ptr = self.head

        # Move the fast pointer 'k' nodes ahead
        count = 0
        while count < k:
            if fast_ptr is None:
                print("Invalid k. Linked list length is less than k.")
                return
            fast_ptr = fast_ptr.next
            count += 1

        # Move both pointers until the fast pointer reaches the end
        while fast_ptr is not None:
            slow_ptr = slow_ptr.next
            fast_ptr = fast_ptr.next

        # If slow_ptr is not None, it is pointing to the kth to the last element
        if slow_ptr is not None:
            return slow_ptr.data
        else:
            print("Invalid k. Linked list length is less than k.")
            return


if __name__ == "__main__":
    # Create an instance of the LinkedList class
    llist = LinkedList()

    # Get space-separated numbers from the user and create a linked list
    elements = input("Enter space-separated numbers to create a linked list:
").split()
    for element in elements:
        llist.push(int(element))

    # Get the value of k from the user
    k = int(input("Enter the value of k: "))

    # Get the kth to the last element from the linked list
    kth_to_last = llist.getKthToLast(k)

    # Print the kth to the last element if it exists
    if kth_to_last is not None:
        print("The kth to the last element is:", kth_to_last)
```

Explanation:

1. **We define two classes:** Node and LinkedList. The Node class represents each element in the linked list, and the LinkedList class represents the collection of those nodes.

2. The LinkedList class has a special function called __init__ that sets the head of the linked list to None when the list is created.

3. The LinkedList class has a method called push that adds a new element to the end of the linked list. It creates a new node with the given data and attaches it to the last node in the list.

4. The LinkedList class has a method called getKthToLast that finds and returns the kth to the last element in the linked list. It takes the value of k as input.

5. In the getKthToLast method, we have two pointers: ptr and kth_ptr. We start both pointers from the head of the list. We move the ptr pointer k steps ahead of the kth_ptr pointer.

6. Then, we move both pointers together until the ptr pointer reaches the end of the list. This way, the kth_ptr pointer will be at the kth to the last element when the traversal is complete.

7. After the traversal, we check if the length of the list is less than k. If it is, we print an error message and return. Otherwise, we return the value of the kth_ptr node.

8. In the driver's code, we create an instance of the LinkedList class. We ask the user to input space-separated numbers to create the linked list, and we add those numbers to the list using the push method.

9. Then, we ask the user to input the value of k. We call the getKthToLast method to find the kth to the last element in the list. If it exists, we print the value.

The loop can be minimized by using a single pointer ptr to traverse the linked list while keeping a kth_ptr initially at the head. We increment both pointers simultaneously until ptr reaches the end of the list. Once the count of traversed nodes is greater than or equal to k, we move the kth_ptr forward. This way, we find the kth to the last element using a single pass .

I was knowing to find the nth number but not kth element from the last, but I tried by creating the linked list in the reverse order. I referred this website and analyzed the code
https://www.techiedelight.com/find-kth-node-from-the-end-linked-list/