Problem 5: Given the head of a linked list, return *the node where the cycle begins. If there is no cycle, return* null.

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


def detectCycle(head):
    if not head or not head.next:
        return None

    slow = head
    fast = head
    while fast and fast.next:
        slow = slow.next
        fast = fast.next.next
        if slow == fast:
            break
    else:

        return None

    fast = head
    while slow != fast:
        slow = slow.next
        fast = fast.next

    return slow

def createLinkedList(values):
    head = None
    current = None
    for val in values:
```

```
        node = ListNode(val)
        if not head:
            head = node
            current = head
        else:
            current.next = node
            current = node
    return head


values = [1, 2, 3, 4, 3, 6, 10]
head = createLinkedList(values)
result = detectCycle(head)
if result:
    print("tail connects to node index", values.index(result.val))
else:
    print("No cycle found.")
```