QUESTION ONE:

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

def hasCycle(head: ListNode) -> bool:
    visited = set()
    while head:
        if head in visited:
            return True
        visited.add(head)
        head = head.next
    return False
```

Input: head = [2,7,4,6,7,8,9], pos = -1
Output: true
Explanation: There is a cycle in the linked list, where the tail connects to the 0^{th} node.

QUESTION TWO:

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

def detectCycleStart(head: ListNode) -> ListNode:
    slow, fast = head, head
    has_cycle = False

    # Detect cycle
    while fast and fast.next:
        slow = slow.next
```

```
        fast = fast.next.next
        if slow == fast:
            has_cycle = True
            break

    # If no cycle, return None
    if not has_cycle:
        return None

    # Move one pointer back to head
    slow = head

    # Move both pointers until they meet again
    while slow != fast:
        slow = slow.next
        fast = fast.next

    return slow

# Example usage
# Create a linked list with a cycle (tail connects to node index 1)
node4 = ListNode(-4)
node3 = ListNode(0, node4)
node2 = ListNode(2, node3)
node1 = ListNode(3, node2)
node4.next = node2                              # Create the cycle

start_of_cycle = detectCycleStart(node1)
if start_of_cycle:
    print("Cycle starts at node with value {start_of_cycle.val}")
else:
    print("No cycle")
```

Input: head = [2,7,4,6,7,8,9], pos = -1
Output: true
Explanation: There is a cycle in the linked list, where the tail connects to the first node.

QUESTION THREE:

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

def reverse_linked_list(head: ListNode) -> ListNode:
    prev = None
    current = head

    while current:
        next_node = current.next
        current.next = prev
        prev = current
        current = next_node

    return prev

# Example usage
# Create a linked list: 1 -> 2 -> 3 -> 4
node4 = ListNode(4)
node3 = ListNode(3, node4)
node2 = ListNode(2, node3)
node1 = ListNode(1, node2)

# Reverse the linked list
reversed_head = reverse_linked_list(node1)
```

```
# Print the reversed list
while reversed_head:
    print(reversed_head.val, end=" -> ")
    reversed_head = reversed_head.next
print("None")
```

Input: head = [2,7,4,6,7,8,9]
Output: [9,8,7,6,4,7,2]
Explanation: The above code defines a “ListNode” class and a function “reverse_linked_list” that takes the head of a linked list and returns the reversed list. It iterates through the list, reversing the pointers between nodes.