**QUESTION 1**

class Solution:

def hasCycle(self, head: Optional[ListNode]) -> bool:

if head is None:

return False

tortoise = head

hare = head

while hare is not None and hare.next is not None:

tortoise = tortoise.next

hare = hare.next.next

if tortoise == hare:

return True

return False

**QUESTION 2**

class Solution:

def detectCycle(self, head: Optional[ListNode]) -> Optional[ListNode]:

if head is None:

return None

tortoise = head

hare = head

while hare is not None and hare.next is not None:

tortoise = tortoise.next

hare = hare.next.next

if tortoise == hare:

break

if hare is None:

return None

# Find the start of the cycle

pointer1 = head

pointer2 = hare

while pointer1 != pointer2:

pointer1 = pointer1.next

pointer2 = pointer2.next

# Find the length of the cycle

length = 0

while pointer2.next != pointer1:

pointer2 = pointer2.next

length += 1

# Reverse the cycle

prev = None

current = pointer1

for \_ in range(length):

next\_node = current.next

current.next = prev

prev = current

current = next\_node

return pointer1

**QUESTION 3**

class Solution:

def reverseList(self, head: Optional[ListNode]) -> Optional[ListNode]:

if head is None:

return None

stack = []

current = head

while current is not None:

stack.append(current)

current = current.next

new\_head = stack.pop()

current = new\_head

while stack:

next\_node = current.next

current.next = stack.pop()

current = next\_node

return new\_head