Question (One) 1

Given head, the head of a linked list, determines if the linked list has a cycle in it. There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to. Note that pos is not passed as a parameter.

Return true if there is a cycle in the linked list. Otherwise, return false.

```
Class solution:
    def hasCycle(self, head: listnode)-> bool:
        slow,fast = head,head
     while fast and fast.next:
       slow = slow.next
       fast = fast.next.next
       if slow == fast:
         return True
     return False
```

QUESTION TWO (2):

Given the head of a linked list, return the node where the cycle begins. If there is no cycle, return null. There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to (0-indexed). It is -1 if there is no cycle. Note that pos is not passed as a parameter.

Do not modify the linked list.

```
class ListNode:
   def _init_(self, x):
      self.val = x
      self.next = None


def detectCycle(head):
   slow = fast = head
   while fast and fast.next:
```

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

    if not fast or not fast.next:
        return None

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

    return slow
```

QUESTION THREE (3):

Write a function that takes the head of a linked list and returns the reversed list.

```
class solution :
  def reverselist(self , head : listnode)  -> listnode:
    prev , curr = None , head
    while curr :
         nxt = curr.next
         curr.next= prev
          prev = curr
          curr = nxt
```

return prev