# Segregate nodes based on pivot

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


def getElementsLessThan(head):
    if head is None or head.next is None:
        return head
    target = getTail(head).data
    dummySmall = Node(-1)
    prevSmall = dummySmall
    dummyLarge = Node(-1)
    prevLarge = dummyLarge
    curr = head
    while curr!=None:
        if curr.data<=target:
            prevSmall.next = curr
            prevSmall = curr
            curr = curr.next
        else:
            prevLarge.next = curr
            prevLarge = curr
            curr = curr.next

    if prevSmall.next!=None:
        prevSmall.next = None
    if prevLarge.next!=None:
        prevLarge.next = None
    prevSmall.next  = dummyLarge.next
    return dummySmall.next




def getTail(head):
    if head.next is None:
        return head
    curr = head
```

```
    while curr.next!=None:
        curr = curr.next
    return curr




head1 = Node(2)
head1.next = Node(1)
head1.next.next = Node(-8)
head1.next.next.next = Node(-5)
head1.next.next.next.next = Node(-12)
head1.next.next.next.next.next = Node(10)
head1.next.next.next.next.next.next = Node(0)

# head2 = Node(7)
# head2.next = Node(8)
# head2.next.next = Node(9)

head = getElementsLessThan(head1, )

while head != None:
    print(head.data, end=' ')
    head = head.next
```