Count triplets in a sorted doubly linked list whose sum is equal to a given value x

Given a sorted doubly linked list of distinct nodes(no two nodes have the same data) and a value **x**. Count triplets in the list that sum up to a given value **x**.

```
def countPairs(first, second, value):

    count = 0

    # The loop terminates when either of two pointers
    # become None, or they cross each other (second.next
    # == first), or they become same (first == second)
    while (first != None and second != None and
           first != second and second.next != first):

        # Pair found
        if ((first.data + second.data) == value):

            # Increment count
            count += 1

            # Move first in forward direction
            first = first.next

            # Move second in backward direction
            second = second.prev

        # If sum is greater than 'value'
        # move second in backward direction
        elif ((first.data + second.data) > value):
            second = second.prev

        # Else move first in forward direction
        else:
            first = first.next

    # Required count of pairs
    return count
```

```
# Function to count triplets in a sorted
# doubly linked list whose sum is equal
# to a given value 'x'
def countTriplets(head, x):

    # If list is empty
    if (head == None):
        return 0

    current, first, last = head, None, None
    count = 0

    # Get pointer to the last node of
    # the doubly linked list
    last = head

    while (last.next != None):
        last = last.next

    # Traversing the doubly linked list
    while current != None:

        # For each current node
        first = current.next

        # count pairs with sum(x - current.data) in
        # the range first to last and add it to the
        # 'count' of triplets
        count, current = count + countPairs(
            first, last, x - current.data), current.next

    # Required count of triplets
    return count

# A utility function to insert a new node
# at the beginning of doubly linked list
def insert(head, data):

    # Allocate node
    temp = Node(data)

    # Put in the data
    # temp.next = temp.prev = None
```

```
    if (head == None):
        head = temp
    else:
        temp.next = head
        head.prev = temp
        head = temp

    return head

# Driver code
if __name__ == '__main__':

    # Start with an empty doubly linked list
    head = None

    # Insert values in sorted order
    head = insert(head, 9)
    head = insert(head, 8)
    head = insert(head, 6)
    head = insert(head, 5)
    head = insert(head, 4)
    head = insert(head, 2)
    head = insert(head, 1)

    x = 17

    print("Count = ", countTriplets(head, x))
```