Implement the merge sort algorithm using a linked list instead of arrays. You can use any kind of a linked structure, such as single, double, circular lists, stacks and/or queues. You can populate your list from an explicitly defined array in your program.

HINT: You will not be using low, middle and high anymore. For finding the middle point, traverse through the linked list while keeping count of the number of nodes. Break up the list into two, null terminated lists, based on the node count.

//Here is a portion of what mergeSort function would look like:

//a points to the left partition, b points to the right partition. They would be passed by reference.

//You would then recursively process each partition.

if(head==NULL)

return NULL;

if (head->next==NULL)

return head;

a=mergeSort(a);

b=mergeSort(b);

c=merge(a,b);

return (c);

//These are the function headers

void split (node\* head,node\*&a,node\*&b)

node\* merge(node\* a, node\* b)

node\* mergeSort( node\* head)

Make sure to account for when head is null or when there is only one item in the list