MERGE SORT FOR LINKED LIST

Given Pointer/Reference to the head of a linked list, task is to Sort the given linked list using Merge Sort.

You need to complete the function splitList() and mergeList(), which will be called by the function mergeSort().

void mergeSort(struct node\*\* headRef)

{

struct node\* head = \*headRef;

struct node\* a, b;

if ((head == NULL) || (head->next == NULL))

return NULL;

splitList(head, &a, &b);

mergeSort(&a);

mergeSort(&b);

\*headRef = mergeList(a, b);

}

The spliList() function takes three input, head of the linked list and references to two pointers that are initially null. The function changes these pointers so that will point to the two new splitted lists, left and right halves respectively. The function splits the linked list in to two halves just like in standard merge sort and store the two new head in pointer a and b respectively. As the name suggests function mergeList will merge two linked lists, as in standard merge sort and should head pointer to of the new merged list.

Note: If the length of linked list is odd, then the extra node should go in the first list while splitting.

Input:

There will be multiple test cases, for each test case function mergeSort() will be called separately. The only input to the function is the pointer/reference to the head of the linked list.

Output:

For each test, print the sorted linked list.

Constraints:

1<=T<=100

1<=N<=105

Example:

Input:

2

5

3 5 2 4 1

3

9 15 0

Ouput:

1 2 3 4 5

0 9 15

Note:The Input/Output format and Example given are used for system's internal purpose, and should be used by a user for Expected Output only. As it is a function problem, hence a user should not read any input from stdin/console. The task is to complete the function specified, and not to write the full code.

**PROGRAM**

/\*Please note that it's Function problem i.e.

you need to write your solution in the form of Function(s) only.

Driver Code to call/invoke your function is mentioned above.\*/

//User function Template for Java

class GfG

{

public static Node sortedMerge(Node a, Node b)

{

Node result=null;

if (a == null)

return b;

if (b == null)

return a;

if (a.data <= b.data)

{

result = a;

result.next = sortedMerge(a.next, b);

}

else

{

result = b;

result.next = sortedMerge(a, b.next);

}

return result;

}

public static Node getMiddle(Node h)

{

if (h == null)

return h;

Node fastptr = h.next;

Node slowptr = h;

while (fastptr != null)

{

fastptr = fastptr.next;

if(fastptr!=null)

{

slowptr = slowptr.next;

fastptr=fastptr.next;

}

}

return slowptr;

}

}

OUTPUT

Correct Answer.Correct Answer  
Execution Time:0.5