Split a Circular Linked List into two halves

Given a **C**irular **L**inked **L**ist of size **N,** split it into two halves circular lists. If there are odd number of nodes in the given circular linked list then out of the resulting two halved lists, first list should have one node more than the second list. The resultant lists should also be circular lists and not linear lists.

Example 1:

```
Input: Circular LinkedList: 1->5->7
Output: 1 5
7
```

Example 2:

```
Input: Circular LinkedList: 2->6->1->5
Output: 2 6
1 5
```

Your Task:
Your task is to complete the given function **splitList**(), which takes 3 input parameters: The address of the head of the linked list, addresses of the head of the first and second halved resultant lists and Set the **head1_ref** and **head2_ref** to the first resultant list and second resultant list respectively.

Expected Time Complexity: O(N)
Expected Auxilliary Space: O(1)

Constraints:
1 <= N <= 100

```
class Solution:
    def splitList(self, head, head1, head2):
        #code here
        if head is None or head.next is None:
            return head
        curr = head
        while curr.next!=head:
            curr = curr.next
        curr.next = None
        fast = head
        slow = head

        while fast.next!=None and fast.next.next!=None:
            slow = slow.next
```

```
            fast = fast.next.next
        newHead = slow.next
        slow.next = None
        slow.next = head
        head1 = head
        temp = newHead
        while temp.next!=None:
            temp = temp.next
        temp.next=newHead
        head2 = newHead
        #this is to emulate pass by reference in python please don't delete
below line.
        return head1,head2
```