# 2674. Split a Circular Linked List

## Description

Given a **circular linked list** `list` of positive integers, your task is to split it into 2 **circular linked lists** so that the first one contains the **first half** of the nodes in `list` (exactly `ceil(list.length / 2)` nodes) in the same order they appeared in `list`, and the second one contains **the rest** of the nodes in `list` in the same order they appeared in `list`.

Return *an array answer of length 2 in which the first element is a* ***circular linked list*** *representing the* ***first half*** *and the second element is a* ***circular linked list*** *representing the* ***second half***.

A **circular linked list** is a normal linked list with the only difference being that the last node's next node, is the first node.

**Example 1:**

```
Input: nums = [1,5,7]
Output: [[1,5],[7]]
Explanation: The initial list has 3 nodes so the first half would be the first 2 elements since ceil(3 / 2) = 2 and the rest which is 1 node is in
the second half.
```

**Example 2:**

```
Input: nums = [2,6,1,5]
Output: [[2,6],[1,5]]
Explanation: The initial list has 4 nodes so the first half would be the first 2 elements since ceil(4 / 2) = 2 and the rest which is 2 nodes are
in the second half.
```

**Constraints:**

- The number of nodes in `list` is in the range $[2, 10^5]$
- $0 <= \text{Node.val} <= 10^9$
- `LastNode.next = FirstNode` where `LastNode` is the last node of the list and `FirstNode` is the first one